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 $O(n/k)$ 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 $O(n/k)$ 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
## Contents

1 Introduction
   1.1 Main Results ........................................... 3

2 Sequential Algorithms for Facility Location with Outliers
   2.1 Robust Facility Location .................................. 6
   2.2 Facility Location with Penalties .......................... 13

3 Distributed Robust Facility Location: Implicit Metric
   3.1 The $k$-Machine Algorithm ................................ 16
      3.1.1 Radius Computation .................................. 16
      3.1.2 Greedy Phase ......................................... 17
      3.1.3 Analysis of the Algorithm ............................ 21
   3.2 The Congested Clique and MPC Algorithms ............... 25

4 Distributed Robust Facility Location: Explicit Metric
   4.1 The Congested Clique Algorithm .......................... 26
   4.2 The MPC Algorithm ......................................... 28

5 Facility Location with Penalties: Implicit Metric
   5.1 The $k$-Machine Algorithm ................................ 28
      5.1.1 Radius Computation .................................. 28
      5.1.2 Analysis of the Algorithm ............................ 31
   5.2 The Congested Clique and MPC Algorithms ............... 34

6 Facility Location with Penalties: Explicit Metric
   6.1 The Congested Clique Algorithm .......................... 35
   6.2 The MPC Algorithm ......................................... 35

7 Conclusion and Open Questions


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 $\text{cost}(F') := \sum_{i \in F'} f_i + \sum_{j \in C} d(j, F')$ 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 $\alpha$-approximation algorithm for FacLoc finds in polynomial time, a subset $F' \subseteq F$ of facilities such that $\text{cost}(F') \leq \alpha \cdot \text{cost}(F^*)$, 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 [15, 16] for clustering algorithms in systems such as MapReduce [12] 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

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

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,

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

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.\cite{31} and then extensively studied in recent years\cite{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31}. 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\cite{29}.

Massively Parallel Computation (MPC) model: The MPC model was introduced in\cite{27} and variants of this model were considered in\cite{2, 3, 20}. 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
Throughout the paper, we use $\tilde{O}(n)$ 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 $\polylog(n)$ bits.

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 [22, 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 $\polylog(n)$ 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 [32].

---

1 Throughout the paper, we use $\tilde{O}(f(n))$ as a shorthand for $O(f(n) \cdot \polylog(n))$ and $\tilde{\Omega}(f(n))$ as a shorthand for $\Omega(f(n)/\polylog(n))$. 
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 [4, 8, 21] and in [16] 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 [4, 21] 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 $O(poly(log\ n))$ 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 $\tilde{O}(n/m)$ 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.

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, [8] 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.

For a facility $i \in F$ and a client $j \in C$, we use the shorthand $c_{ij} := d(i, j)$. Also, for a facility $i \in F$ and a radius $r \geq 0$, let $B(i, r)$ denote the set of clients within the distance $r$, i.e., $B(i, r) := \{j \in C \mid c_{ij} \leq r\}$.
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 \(i_e \in F\) be a facility in that solution with the maximum opening cost \(f_{i_e}\). We begin by assuming that we are given a facility, say \(i_e\) with opening cost \(f_{i_e}\), such that, \(f_{i_e} \leq f_i \leq \alpha f_{i_e}\), 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 \(\text{cost}_e(C^*_e, F^*_e)\) 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.

\textbf{Lemma 1.} \(\text{cost}_e(C^*_e, F^*_e) \leq \text{cost}(C^*, F^*)\).

\textbf{Proof.} Recall that \(i_e\) satisfies \(f_{i_e} \leq f_i \leq \alpha f_{i_e}\), where \(i_e\) is the facility with largest opening cost in \(F^*\). In the modified instance, all facilities with opening cost greater than \(f_{i_e}\) are removed; however, no facility from \(F^*\) is removed, because \(f_{i_e} \leq f_i\). Therefore, \((C^*_e, F^*_e)\) is a feasible solution for the modified instance. This implies \(\text{cost}_e(C^*_e, F^*_e) \leq \text{cost}_e(C^*, F^*)\), which follows from the optimality of \((C^*_e, F^*_e)\). Finally, recall that for any facility \(i \in F^*, f'_i = f_i\), and hence \(\text{cost}_e(C^*_e, F^*_e) = \text{cost}(C^*, F^*)\).

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

\begin{algorithm}
\caption{\textsc{FacilityLocationMP}(F, C)}

\begin{algorithmic}
\State /* Radius Computation Phase: */
\State For each \(i \in F\), compute \(r_i \geq 0\), satisfying \(f_i = \sum_{j \in C} \max\{0, r_i - c_{ij}\}\).
\State /* Greedy Phase: */
\State Sort and renumber facilities in the non-decreasing order of \(r_i\).
\State \(F' \leftarrow \emptyset\) \quad \triangleright \text{Solution set}
\For {\(i = 1, 2, \ldots\)}
\If {there is no facility in \(F'\) within distance \(2r_i\) from \(i\)}
\State \(F' \leftarrow F' \cup \{i\}\)
\EndIf
\EndFor
\State Connect each client \(j\) to its closest facility in \(F'\).
\end{algorithmic}
\end{algorithm}
Consider,

If |
\[\text{cost}(C'_e^i, F'_e^i) \leq \text{cost}_e(C'_e^i, F'_e^i) + f_i \quad \text{(if } i \in F'_e^i)\]

\[\leq \beta \cdot \text{cost}_e(C'_e^i, F'_e^i) + (\gamma + 1) \cdot f_i \]

\[\leq \beta \cdot \text{cost}(C^*, F^*) + (\gamma + 1) \cdot f_i \quad \text{(From lemma 4)}\]

\[\leq \beta \cdot \text{cost}(C^*, F^*) + \alpha \cdot (\gamma + 1) \cdot f_i \quad \text{(if } f_i \leq \alpha f_i)\]

\[\leq \beta \cdot \text{cost}(C^*, F^*) + \alpha \cdot (\gamma + 1) \cdot \text{cost}(C^*, F^*) \quad \text{where some clients } \tilde{O} \text{ again, in the outlier determination phase} \]

To efficiently find a facility \( i_e \) satisfying \( f_{i_e} \leq f_{i_e} \leq \alpha f_{i_e} \), we partition the facilities into sets where each set contains facilities with opening costs from the range \([(1 + \varepsilon)^i, (1 + \varepsilon)^{i+1})\). 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, \( f_{i_e} \leq f_{i_e} \leq (1 + \varepsilon)f_{i_e} \). The total number of such iterations will be \( O(\log_{1+\varepsilon} f_{\min}) \), 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 \( \varepsilon \) 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 4), 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 \((C'_t^i, F'_t^i)\) 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 \( t \).

Moreover, we will consider the facility costs \( f' \) to be the modified facility opening costs in the same iteration \( t \). We can ignore the facilities with opening cost \(+\infty\) and add \( i_e \) to any solution with no additional cost. Therefore in our analysis, we just ignore these facilities and use the original facility costs \( f \) for other facilities since they are the same as the modified costs.

Note that we exit the greedy phase if we either process all facilities or we break at line 13, each of which corresponds to the cases (i) \(|O'| > \ell\) where some outliers become clients and (ii) \(|O'| < \ell\) where some clients become outliers again, in the outlier determination phase (we are done if \(|O'| = \ell\)).

Let \( C'' \) and \( O'' \) denote the sets \( C' \) and \( O' \) just before the outlier determination phase. Note that we exit the greedy phase if we either process all facilities or we break at line 13 each of which corresponds to the cases (i) \(|O'| > \ell\) and (ii) \(|O'| < \ell\) in the outlier determination phase (we are done if \(|O'| = \ell\)).

For a client \( j \in C' \), let \( v'_j := \min_{i \in F'} \max\{r_i, c_{ij}\} \). To make the analysis easier, we consider a more expensive solution \((\tilde{C}', F')\) where the set of clients \( \tilde{C}' \) is constructed using the following modified outlier determination phase:

1. If \(|O''| > \ell\), then let \( O_1 \subseteq O'' \) be a set of \(|O''| - \ell\) clients that have the smallest \( v'_j \) values.

In this case, we let \( \tilde{C}' \leftarrow C'' \cup O_1, \quad \tilde{O}' \leftarrow O'' \setminus O_1 \).

\[\begin{align*}
\text{Proof.} \quad & \text{Consider,} \\
\text{cost}(C'_e^i, F'_e^i) & \leq \text{cost}_e(C'_e^i, F'_e^i) + f_i \\
& \leq \beta \cdot \text{cost}_e(C'_e^i, F'_e^i) + (\gamma + 1) \cdot f_i \\
& \leq \beta \cdot \text{cost}(C^*, F^*) + (\gamma + 1) \cdot f_i \quad \text{(From lemma 4)} \\
& \leq \beta \cdot \text{cost}(C^*, F^*) + \alpha \cdot (\gamma + 1) \cdot f_i \quad \text{(if } f_i \leq \alpha f_i) \\
& \leq \beta \cdot \text{cost}(C^*, F^*) + \alpha \cdot (\gamma + 1) \cdot \text{cost}(C^*, F^*) \quad \text{where some clients } \tilde{O} \text{ again, in the outlier determination phase} \\
\end{align*}\]

\[\boxdot\]
Algorithm 2: ROBUSTFACLOC($F, C, p$)

/* Recall: $\ell := |C| - p$ */
for $t = 0, \ldots, O(\log n)$ do
  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 $\varepsilon > 0$
  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: */
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: */
Sort and renumber facilities in the non-decreasing order of $r_i$.
Let $C' \leftarrow \emptyset$, $F' \leftarrow \emptyset$, $O' \leftarrow C$
Let $F_0 \leftarrow \emptyset$
for $i = 1, 2, \ldots$ do
  if there is no facility in $F'$ within distance $2r_i$ from $i$ then
    $F' \leftarrow F' \cup \{i\}$
  end
  $F_i \leftarrow F_{i-1} \cup \{i\}$
  Let $C_i$ denote the set of clients that are within distance $r_i$
  $C' \leftarrow C' \cup C_i$, $O' \leftarrow O' \setminus C_i$. 
  if $|O'| \leq \ell$ then break
end
/* Outlier Determination Phase: */
if $|O'| > \ell$ then
  Let $O_1 \subseteq O'$ be a set of $|O'| - \ell$ clients that are closest to facilities in $F'$. 
  $C' \leftarrow C' \cup O_1$, $O' \leftarrow O' \setminus O_1$.
end
else if $|O'| < \ell$ then
  Let $O_2 \subseteq C'$ be the set of $\ell - |O'|$ clients with largest distance to open facilities $F'$.
  $C' \leftarrow C' \setminus O_2$, $O' \leftarrow O' \cup O_2$.
end
Let $(C'_t, F'_t) \leftarrow (C', F')$
return $(C'_t, F'_t)$ with the minimum cost.
Otherwise, if $|O'| < \ell$. Let $O_2 \subseteq C_i$ be the set of clients with largest $\ell - |O'|$ $v_j'$-values ($i$ is last iteration). In this case, we let $C' \leftarrow C' \setminus O_2$, $O' \leftarrow O' \cup O_2$.

It is easy to see by an exchange argument that the cost$(C', F') \leq$ cost$(\bar{C}', \bar{F}')$, the outliers determined in the algorithm are at least as far from $F'$ as ones in the modified outlier determination phase. Henceforth, we analyze the cost of the solution $(\bar{C}', \bar{F}')$ by comparing it to the cost of a feasible dual LP solution and in order to alleviate excessive notation, we will henceforth refer to the solution $(\bar{C}', \bar{F}')$ as $(C', F')$ and $O'$ as $O'$. We state the standard primal and dual linear programming relaxations for the Robust FacLoc problem in Figure 1.

Now, we construct a feasible dual solution $(v, w, q)$.

For a facility $i \in F$ and client $j \in C$, let $w_{ij} := \max \{0, r_i - c_{ij}\}$. Let $q := \max_{j \in C'} v_j'$ (recall that $v_j' := \min_{i \in F} \max \{r_i, c_{ij}\} = \min_{i \in F} c_{ij} + w_{ij}$). Now, for a client $j \in C$, define $v_j$ as follows:

$$v_j = \begin{cases} v_j' & \text{if } j \in C' \\ q & \text{if } j \in O' \end{cases}$$

Claim 3. A client $j \in C_i$ iff $v_j' \leq r_i$.

Proof. Client $j$ is added to $C_i$ iff for some $i' \in F_i$, $j \in B(i', r_i)$. This means $c_{ij} \leq r_i$ and $r_{i'} \leq r_i$ since we process facilities in increasing order of $r$-value. This means $v_j' \leq \max \{r_{i'}, c_{ij}\} \leq r_i$.

Lemma 4. The solution $(v, w, q)$ is a feasible solution to the dual LP relaxation.

Proof. Note that constraints (6) and (8) of the dual are satisfied by construction and so is constraint (5) for clients $j \in C'$. Therefore, in order to show that the solution $(v, w, q)$ is feasible, we have to show that constraint (5) is satisfied for all clients $j \in O'$. To this end, we consider the following two cases.

Case 1. We enter the outlier determination phase after iterating over all facilities in $F$. Therefore, we have $|O'| > \ell$. This means that we identified a set $O_1 \subset O'^*$ of size $|O'^*| - \ell$ to be marked as non-outliers.

As we iterate over all the facilities, if $j \in O'^*$ then by claim (4) $v_j' > \max_{i \in F} r_i \geq \max_{j \in C'} v_j'$.

We put in $O_1$ the clients from $O'^*$ that have smallest $v_j'$-values. This means that for all $j \in O'$, $v_j' \geq \max_{j \in O_1} v_j' = \max_{j \in C'} v_j' = q$ where the second equality is because for $v_j'$ for $j \in C'$ is at most $v_j'$ for $j' \in O_1$. 

### Figure 1

**Primal and Dual Linear Programming Relaxations for Robust FacLoc**

| Primal LP | Dual LP |
|-----------|---------|
| $\text{minimize } \sum_{i \in F} f_i y_i + \sum_{i \in F, j \in C} c_{ij} x_{ij}$ | $\text{maximize } \sum_{j \in C} v_j - \ell \cdot q$ |
| subject to $z_j + \sum_{i \in F} x_{ij} \geq 1$, $\forall j \in C$ (1) | subject to $v_j \leq c_{ij} + w_{ij}$, $\forall j \in C$ (5) |
| $x_{ij} \leq y_i$, $\forall i \in F, \forall j \in C$ (2) | $\sum_{j \in C} w_{ij} \leq f_i$, $\forall i \in F$ (6) |
| $\sum_{j \in C} z_j \leq \ell$, (3) | $v_j \leq q$, $\forall j \in C$ (7) |
| $r_j, y_i, x_{ij} \in [0, 1]$, $\forall i \in F, \forall j \in C$ (4) | $v_j, w_{ij} \geq 0$ $\forall i \in F, v_j \in C$ (8) |

Proof. Client $j$ is added to $C_i$ iff for some $i' \in F_i$, $j \in B(i', r_i)$. This means $c_{ij} \leq r_i$ and $r_{i'} \leq r_i$ since we process facilities in increasing order of $r$-value. This means $v_j' \leq \max \{r_{i'}, c_{ij}\} \leq r_i$.

Lemma 4. The solution $(v, w, q)$ is a feasible solution to the dual LP relaxation.
Therefore, we conclude that for any \( j \in O' \) and \( i \in F \), \( c_{ij} + w_{ij} \geq v'_j \geq q = v_j \).

**Claim 5.** We enter the outlier determination phase because of the break statement on line 114. Here, \( |O''| \leq \ell \) and \( C' \subseteq C'' \).

Let \( i^* \) be the last iteration of the for loop. Therefore \( F_i' \) is the set of facilities we consider in the for loop.

Recall that by the case assumption we have \( |O''| \leq \ell \) and therefore \( C' \subseteq C'' \). All clients \( j \in C'' \) were part of \( C_i \) for some \( i \in F_i' \), and by claim 3 we have \( v'_j \leq r_i \leq r_i \). Therefore,

\[
q = \max_{j \in C''} v_j = \max_{j \notin C'} v_j \leq r_i.
\]

Let \( j \in O' \) be an outlier client. If \( j \in O'' \), then for any facility \( i \in F' \),

\[
c_{ij} + w_{ij} \geq v'_j \geq r_i, \quad \text{(Otherwise } j \text{ would be in } C'' \text{ by claim 3)}
\]

\[
\geq q = v_j
\]

Otherwise, \( j \in O_2 \) and was added to \( O' \) because it had highest \( v'_j \) value among \( C_i \).

Therefore, it follows that for any facility \( i \in F \), \( c_{ij} + w_{ij} \geq v'_j \geq \max_{j \in C'} v'_j = q = v_j \).

From the case analysis it follows that \( v_j \leq c_{ij} + w_{ij} \) for all \( j \in O' \) and for all \( i \in F \). Therefore, we have shown that \((v, w, q)\) is a dual feasible solution.

For the approximation guarantee we can focus just on the clients in \( C' \) because the only contribution that the clients in \( O' \) make to the dual objective function is to cancel out the \(-\ell q\) term and hence they do not affect the approximation guarantee. We call a facility \( i \) the bottleneck of \( j \) if \( v'_j = \max\{r_i, c_{ij}\} = c_{ij} + w_{ij} \).

We first prove a few straightforward claims about the dual solution.

**Claim 5.** For any \( i \in F' \) and \( j \in C' \), \( r_i \leq c_{ij} + w_{ij} \). Moreover if \( w_{ij} > 0 \) then \( r_i = c_{ij} + w_{ij} \).

**Proof.** \( w_{ij} = \max\{0, r_i - c_{ij}\} \geq r_i - c_{ij} \). Now if \( w_{ij} > 0 \) then \( r_i > c_{ij} \) and we have \( w_{ij} = r_i - c_{ij} \) which implies the claim.

**Claim 6.** If \( \iota \) is a bottleneck for \( j \in C' \), then \( v_j \geq r_i \).

**Proof.** For \( j \in C' \), \( v_j = v'_j = \max\{c_{ij}, r_i\} \geq r_i \).

**Claim 7.** If \( \iota \in F' \) is a bottleneck for \( j \in C' \), then \( w_{ij} = 0 \) for all \( i' \in F' \), where \( i' \neq \iota \).

**Proof.** Assume for contradiction that \( w_{i'j} > 0 \), i.e., \( r_{i'} = c_{i'j} \).

If \( r_{i'} \geq c_{ij} \), then \( v_j = r_i \leq c_{ij} + w_{ij} = r_{i'} \). In this case, \( c_{i'jr_{i'}} \leq c_{ij} + c_{i'j} \leq 2r_{i'} \).

Otherwise, if \( c_{ij} > r_{i'} \), then \( v_j = c_{ij} \leq r_{i'} \). Here too we have, \( c_{i'j} \leq c_{ij} + c_{i'j} \leq 2r_{i'} \).

In either case, \( c_{i'j} \leq 2r_{i'} \leq 2\max\{r_{i'}, r_i\} \), which is a contradiction, since at most one of \( i', \iota \) can be added to \( F' \).

**Claim 8.** If a closed facility \( \iota \notin F' \) is the bottleneck for \( j \in C' \), and if an open facility \( i' \in F' \) caused \( \iota \) to close, then \( c_{i'j} \leq 3v_j \).

**Proof.** \( \iota \) is a bottleneck for \( j \), so \( v_j \geq c_{ij} \), and \( v_j \geq r_i \).

Since \( i' \in F' \) caused \( \iota \) to close, then \( c_{ki} \leq 2r_i \leq 2v_j \). Therefore, \( c_{i'j} \leq c_{ki} + c_{ij} \leq 3v_j \).

Now we state the main lemma that uses the dual variables for analyzing the cost.

**Lemma 9.** For any \( j \in C' \), there is some \( i \in F' \) such that \( c_{ij} + w_{ij} \leq 3v_j \).
Proof. Fix a client \( j \in C' \), and let \( i \) be its bottleneck. We consider different cases. Here, \( c_{ij} \) should be seen as the connection cost of \( j \), and \( w_{ij} \), the cost towards opening of a facility (if \( w_{ij} > 0 \)).

**Case 1:** \( i \in F' \).

In this case, by claim [8] \( w_{ij} = 0 \) for all other \( i' \in F' \).

If \( c_{ij} < r_i \), then \( w_{ij} > 0 \), and \( v_j = c_{ij} + w_{ij} \). Therefore, \( v_j \) pays for the connection cost as well as towards the opening cost of \( i \).

Otherwise, if \( c_{ij} \geq r_i \), then \( w_{ij} = 0 \). Also, \( w_{ij} = 0 \) for all other \( i' \in F' \). Therefore, \( j \) does not contribute towards opening of any facility in \( F' \). Also, we have \( v_j = \max \{ c_{ij}, r_i \} = c_{ij} \), i.e., \( v_j \) pays for \( j \)'s connection cost.

**Case 2:** \( i \notin F' \) and \( w_{ij} = 0 \) for all \( i \in F' \).

Let \( i' \in F' \) be the facility that caused \( i \) to close. From claim [8] we have that \( c_{ij} \leq 3v_j \), i.e. \( 3v_j \) pays for the connection cost of \( j \).

**Case 3:** \( i \notin F' \), and there is some \( i' \in F' \) with \( w_{ij} > 0 \). But \( i' \) did not cause \( i \) to close.

Since \( w_{ij} > 0 \), by claim [9] \( w_{ij} = r_i - c_{ij} \) and \( r_i > c_{ij} \).

Let \( i \) be the facility that caused \( i \) to close. Therefore, \( c_{ii} \leq 2r_i \). Also, \( c_{ij} \leq c_{ii} + c_{ij} \leq 2r_i + c_{ij} \leq 3v_j \).

Now, since \( i \) and \( i' \) both belong to \( F' \), \( c_{ij} > 2 \max \{ r_i, r_{i'} \} \geq 2r_i = 2(c_{ij} + w_{ij}) \).

Therefore, by triangle inequality, \( c_{ij} + c_{ij} \geq c_{ii} > 2(c_{ij} + w_{ij}) \) which implies \( c_{ij} > c_{ij} + 2w_{ij} \).

It follows that, \( c_{ij} + w_{ij} \leq c_{ij} + 2w_{ij} < c_{ij} \leq 3v_j \). Therefore, \( 3v_j \) pays for the connection cost of \( j \) to \( i' \), and its contribution towards opening of \( i' \).

**Case 4:** \( i \notin F' \), but for \( i' \in F' \) that caused \( i \) to close has \( w_{ij} > 0 \).

Again, since \( w_{ij} > 0 \), by claim [9] \( w_{ij} = r_i - c_{ij} \) and \( r_i > c_{ij} \).

From claim [8] we have that \( v_j \geq r_i \). Then, since \( i' \) caused \( i \) to close, we have \( r_i \geq r_{i'} = c_{ij} + w_{ij} \). This implies \( v_j \geq c_{ij} + w_{ij} \), i.e., \( v_j \) pays for the connection cost of \( j \) to \( i' \), and its contribution towards opening of \( i' \).

Now we are ready to prove the approximation guarantee of the algorithm.

**Theorem 10.** \( \text{cost}_c(C', F') \leq 3 \cdot \text{cost}_c(C^*_c, F^*_c) + f_{i^*} \)

Proof. Recall that \( f_{i^*} \) denotes the cost of the most expensive facility in an optimal solution. Furthermore, notice that for any facility \( i \in F' \setminus \{ i^* \} \), the clients in the ball \( B(i, r_i) \subseteq C' \).

However, if \( i^* \in F' \), some of the clients in \( B(i^*, r_{i^*}) \) may have been removed in the outlier determination phase, and therefore it may not get paid completely by the dual variables \( v_j \).

Therefore,

\[
\text{cost}_c(C', F') = \sum_{j \in C'} d(j, F) + \sum_{i \in F' \setminus \{ i^* \}} f_i + f_{i^*} \\
\leq 3 \cdot \sum_{j \in C'} v_j + f_{i^*} \quad \text{(From lemma [9])} \\
= 3 \cdot \left( \sum_{j \in C} v_j - q\ell \right) + f_{i^*} \quad \text{(For } j \in O', v_j = q_i \text{, and } |O'| = \ell \)
\[
\leq 3 \cdot \left( \sum_{j \in C} v_j - q^f \right) + f_{i_e} \quad \text{(Since } i_e \text{ is the most expensive facility)}
\]

Since \((v, w, q)\) is a feasible dual solution, its cost is a lower bound on the cost of any integral optimal solution. Therefore, the theorem follows. \(\blacksquare\)

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

\textbf{Theorem 11.} 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
\[
f_i = \sum_{j \in C} \max \left\{ \min \{r_i - c_{ij}, p_j - c_{ij}\}, 0 \right\}.
\]
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 \(f_i = \sum_{j \in C} \max \left\{ \min \{r_i - c_{ij}, p_j - c_{ij}\}, 0 \right\}\). The algorithm for FacLoc with Penalties is shown in Algorithm 3.

```
/* Radius Computation Phase: */
1 Compute \(r_i\) for each \(i \in F\) satisfying \(f_i = \sum_{j \in C} \max \left\{ \min \{r_i - c_{ij}, p_j - c_{ij}\}, 0 \right\}\).

/* 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    if \(c_{ij} \leq p_j\) then \(C' \leftarrow C' \cup \{j\}\)
11    else \(O' \leftarrow O' \cup \{j\}\)
12 end
13 return \((C', F')\) as the solution.
```

We state the standard primal and dual linear programming relaxations for FacLoc with Penalties in Figure 2. For \(j \in C\) and \(i \in F\), define \(w_{ij} := \max \left\{ \min \{r_i - c_{ij}, p_j - c_{ij}\}, 0 \right\}\) and for \(j \in C\), let \(v_j := \min_{i \in F} c_{ij} + w_{ij}\). Note that \(v_j = \min_{i \in F} \max \{c_{ij}, \min \{r_i, p_j\}\}\). If \(\iota\)
minimize \[ \sum_{j \in F} f_j y_j + \sum_{i \in F, j \in C} c_{ij} x_{ij} + \sum_{j \in C} p_j z_j \]
subject to \[ z_j + \sum_{i \in F} x_{ij} \geq 1, \quad \forall j \in C \] (9)
\[ x_{ij} \leq y_i, \quad \forall i \in F, \forall j \in C \] (10)
\[ z_j, y_i, x_{ij} \in [0, 1] \quad \forall i \in F, \forall j \in C \] (11)

Primal LP

maximize \[ \sum_{j \in C} v_j \] (12)
subject to \[ v_j \leq c_{ij} + w_{ij}, \quad \forall j \in C \] (13)
\[ \sum_{j \in C} w_{ij} \leq f_i, \quad \forall i \in F \] (14)
\[ v_j \leq p_j, \quad \forall j \in C \] (15)
\[ v_j, w_{ij} \geq 0 \quad \forall i \in F, \forall j \in C \] (16)

Dual LP

\textbf{Figure 2:} Primal and Dual Linear Programming Relaxations for FacLoc with Penalties

is a facility realizing the minimum \( v_j = c_{ij} + w_{ij} = \max \{ c_{ij}, \min \{ r_i, p_j \} \} \), then we say that \( \iota \) is the bottleneck of \( j \).

To make the analysis easier, we consider a more expensive solution \((\tilde{C}', F')\) where the set of clients \( \tilde{C}' \) is constructed using the following modified outlier determination phase: for each client \( j \), if \( \max \{ r_i, c_{ij} \} \leq p_j \) then \( j \in \tilde{C}' \) and otherwise \( j \in \Omega' \) where \( \iota \) is the bottleneck of \( j \).

It is easy to see that for any client \( j \in C \), the “cost” paid by the client (i.e., connection cost, or its penalty) in the solution \((C', F')\) is at most the cost paid by it in the solution \((\tilde{C}', F')\). So henceforth, we analyze the cost of the solution \((\tilde{C}', F')\) by comparing it to the cost of a feasible dual LP solution and in order to alleviate excessive notation, we will henceforth refer to the solution \((\tilde{C}', F')\) as \((C', F')\) and \( \Omega' \) as \( \Omega' \).

Because of the way we choose the outliers in the solution we consider for the analysis \((C', F')\) we have the following property (where \( \iota \) is the bottleneck of \( j \)) –

\[ v_j = \begin{cases} \max \{ c_{ij}, r_i \} & \text{if } j \in C' \\ p_j & \text{if } j \in C \setminus C' \end{cases} \]

We simultaneously prove feasibility of the dual solution we constructed, and show how it can be used to pay for the integral solution. We consider different cases regarding a fixed client \( j \in C \) with bottleneck facility \( \iota \). We first prove a few straightforward claims.

\textbf{Claim 12.} If \( \iota \in F' \) is the bottleneck for \( j \in C \), then \( w_{ij} = 0 \) for all \( i' \in F' \), where \( i' \neq \iota \).

\textbf{Proof.} Suppose there exists a facility \( i' \in F' \) with \( w_{ij} > 0 \). That is, \( \min \{ r_{i'}, c_{ij}, p_j, c_{ij} \} \) > 0, which further implies that \( c_{ij} < \min \{ p_j, r_{i'} \} \leq r_{i'} \).

If \( c_{ij} > r_i \), then \( v_j = c_{ij} + \max \{ 0, \min \{ r_i - c_{ij}, p_j - c_{ij} \} \} = c_{ij} \). However, since \( w_{ij} > 0 \), \( c_{ij} = v_j \leq c_{ij} + w_{ij} = \min \{ p_j, r_{i'} \} \leq r_{i'} \)

Otherwise, \( c_{ij} \leq r_i \).

Therefore in either case, \( c_{ij} \leq c_{ij} + c_{ij} \leq 2 \max \{ r_{i'}, r_i \} \), which is a contradiction since at most one of \( \iota, i' \) can belong to \( F' \).

\textbf{Claim 13.} If \( \iota \notin F' \) is the bottleneck of \( j \in C \) and \( \max \{ r_i, c_{ij} \} \leq p_j \), and \( i' \in F' \) caused \( \iota \) to close, then \( c_{ij} \leq 3 r_{i'} \).

\textbf{Proof.} Note that since we assume \( \max \{ r_i, c_{ij} \} \leq p_j \), we have \( j \in C' \), \( v_j = v_{ij}' \), \( c_{ij} \geq p_j \), and \( r_i \leq p_j \). Since \( i' \in F' \) caused \( \iota \) to close, \( r_{i'} \leq r_i \). Furthermore, \( c_{ij} \leq 2r_i \). Therefore, \( c_{ij} \leq c_{ij} + c_{ij} \leq 2r_i + c_{ij} \).
Lemma 16. First note that constraints 14, 15, and 16 are satisfied by construction for all \( i \). This means 2\( r_i + c_{ij} \leq 3v_j \). Otherwise, \( c_{ij} < r_i \). Here, \( v_j = \min \{ r_i, p_j \} \). Then 2\( r_i + c_{ij} < 3r_i \) is satisfied. So, \( c_{ij} \leq 3v_j \).

In either case, \( c_{ij} \leq 3v_j \).

Claim 14. If \( i \) is the bottleneck of \( j \), with \( \max \{ r_i, c_{ij} \} \leq p_j \), then \( v_j \geq r_i \).

Proof. Again, since we assume \( \max \{ r_i, c_{ij} \} \leq p_j \), if \( j \) has \( v_j = v_j' \), \( c_{ij} \geq p_j \), and \( r_i \leq p_j \).

Recall that \( v_j = v_j' = \max \{ c_{ij}, \min \{ r_i, p_j \} \} = \max \{ c_{ij}, r_i \} \geq r_i \) and the claim follows.

We are now ready to prove the feasibility and approximation guarantee.

Lemma 15. The solution \((v, w)\) is a feasible solution to the dual LP relaxation.

Proof. First note that constraints 14, 15, and 16 are satisfied by construction for all \( i \) and \( j \) and so is constraint 13 for all \( j \) in \( C' \).

All that is left to show is that constraint 13 is satisfied for all \( j \) in \( O' \). Since \( j \) in \( O' \), \( \max \{ r_i, c_{ij} \} > p_j \).

We have, \( v_j := p_j < \max \{ r_i, c_{ij} \} = \max \{ c_{ij}, \min \{ r_i, p_j \} \} = v_j' \leq c_{ij} + w_{ij} \) for any \( i \) in \( F \).

Lemma 16. For any \( j \) in \( C' \), there is some \( i \) in \( F' \) such that \( c_{ij} + w_{ij} \leq 3v_j \).

Proof. In all the cases, we assume that \( \max \{ r_i, c_{ij} \} \leq p_j \) and therefore \( j \) in \( C' \). This also implies \( v_j = v_j' = \max \{ c_{ij}, \min \{ p_j, r_i \} \} = \max \{ c_{ij}, r_i \} \). Therefore, we can just disregard the penalties in the analysis.

Case 1. \( i \in F' \)

Connect \( j \) to \( i \). From claim 13, we know that \( w_{ij} = 0 \) for all other \( i' \) in \( F' \).

1. If \( c_{ij} < r_i \), then \( v_j = c_{ij} + w_{ij} \). In this case, \( v_j \) pays for connecting \( j \) to \( i \) and also for \( j \)'s contribution to opening cost of \( i \) which is exactly \( w_{ij} \).

2. Otherwise, \( c_{ij} \geq r_i \), then \( w_{ij} = 0 \), which is \( j \)'s contribution towards \( i \). We have \( v_j = c_{ij} \) and therefore \( v_j \) pays for connecting \( j \) to \( i \).

Case 2. \( i \notin F' \) and \( w_{ij} = 0 \) for all \( i \) in \( F' \).

Let \( i' \) be the facility that caused \( i \) to close. Connect \( j \) to \( i' \). From claim 13, we have \( c_{ij} \leq 3v_j \). Therefore, \( 3v_j \) pays for the connection to \( i' \).

Case 3. \( i \notin F' \), there is some \( i' \) in \( F' \) with \( w_{ij} > 0 \), but \( i' \) did not cause \( i \) to close.

We connect \( j \) to \( i' \). By assumption \( w_{ij} = r_i - c_{ij} > 0 \). Furthermore, let \( i \) be the facility that caused \( i \) to close. By claim 13, we have \( c_{ij} \leq 3v_j \).

We have \( c_{ij} + w_{ij} = r_i \). Also, \( c_{ij} > 2r_i \), since \( i', i \) both were added to \( F' \).

Now, \( 2(c_{ij} + w_{ij}) = 2r_i \leq c_{ij} + w_{ij} \). Subtracting \( c_{ij} \) from both sides, we get \( c_{ij} + 2w_{ij} \leq c_{ij} \leq 3v_j \). Therefore, \( 3v_j \) pays for the connection cost of \( j \) to \( i' \) and also for (twice) \( j \)'s contribution towards opening \( i' \).

Case 4. \( i \notin F' \) and \( i' \) in \( F' \) with \( w_{ij} > 0 \) caused \( i \) to close.

We connect \( j \) to \( i' \). From claim 13, we have that \( v_j \geq r_i \).

Since \( i' \) caused \( i \) to close, \( r_i \geq r_i \geq c_{ij} + w_{ij} \). Therefore, \( c_{ij} + w_{ij} \leq r_i \). Since \( v \) is \( 3 \), \( v \) pays for the connection cost of \( j \) to \( i' \), as well as its contribution towards opening of \( i' \).
Thus, \((v, w)\) is a feasible dual solution. We use the above analysis to conclude with the following theorem.

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

\[ \textbf{Theorem 17.} \ \text{cost}(C', F') \leq 3 \cdot \text{cost}(C^*, F^*). \]

\[ \textbf{Proof.} \] We show \(\text{cost}(C', F') \leq 3 \left( \sum_{j \in C} v_j \right)\), which is sufficient since \((v, w)\) is a feasible dual solution, and cost of any feasible dual solution is a lower bound on the cost of an integral optimal solution.

As we have argued previously, for any \(j \in C \setminus C'\), we have \(p_j = v_j\), and that for any \(j \in C'\), we have \(d(j, F') + s(j)\), where \(s(j) \geq 0\) is the contribution of \(j\) towards opening a single facility in \(F'\). We have also argued that any \(j \in C'\) contributes \(s(j)\) for at most one open facility from \(F'\). It follows that,

\[
\text{cost}(C', F') = \sum_{i \in F'} f_i + \sum_{j \in C'} d(j, F') + \sum_{j \in C \setminus C'} p_j
\]

\[
= \left( \sum_{j \in C'} s(j) + d(j, F') \right) + \sum_{j \in C \setminus C'} p_j
\]

\[
\leq 3 \sum_{j \in C'} v_j + \sum_{j \in C \setminus C'} v_j
\]

\[
\leq 3 \left( \sum_{j \in C} v_j \right)
\]

\[ \blacksquare \]

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.

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.

\[ \textbf{Lemma 18 (Corollary 1 in [4].)} \] 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 \(O((n/k) \cdot \text{poly}(\log n)/\text{poly}(\varepsilon))\) rounds.

In addition to SSSP, our algorithms require an efficient solution to a more general problem that we call \textit{Multi-Source Shortest Paths} (in short, MSSP) and a variant of MSSP that we
call EXCLUSIVEMSSP. 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. Whereas in EXCLUSIVEMSSP, for each $v \in T$, we are required to output $d(v, T \setminus \{v\})$ and the vertex $u^* \in T \setminus \{v\}$ that realizes this distance. The following two lemmas show that we can solve these two problems efficiently in the $k$-machine model.

> **Lemma 19** (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 $0 \leq \varepsilon \leq 1$, compute a $(1 + \varepsilon)$-approximation to MSSP in $\tilde{O}(1/\text{poly}(\varepsilon) \cdot n/k)$ 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 $d(v, u) \leq \tilde{d} \leq (1 + \varepsilon) \cdot d(v, T)$.

> **Lemma 20** (Lemma 5 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 $0 \leq \varepsilon \leq 1$, compute a $(1 + \varepsilon)$-approximation to EXCLUSIVEMSSP in $\tilde{O}(1/\text{poly}(\varepsilon) \cdot n/k)$ rounds, w.h.p. Specifically, after the algorithm has ended, for each $v \in T$, the machine $m_j$ that hosts $v$ knows a pair $(u, \tilde{d}) \in T \setminus \{v\} \times \mathbb{R}^+$, such that $d(v, u) \leq \tilde{d} \leq (1 + \varepsilon) \cdot d(v, T \setminus \{v\})$.

### 3.1.1 Radius Computation

Using the primitives we described in the previous section, [4] show that it is possible to compute approximate radius values efficiently in the $k$-machine model by computing neighborhood size estimates along the lines of [11, 39]. A version of the algorithm is described in [4]. We discuss the implementation of this algorithm in a fair bit of detail because we will need to modify certain aspects when implementing the FACLOC with Penalties algorithm in the $k$-machine model (Section 5.1.1).

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

**Algorithm 4: RadiusComputation Algorithm**

1. **Neighborhood-Size Computation.** Each machine $m_j$ computes $q_i(v)$, for all integers $i \geq 0$ 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 $t \geq 1$ is the smallest integer for which $\sum_{i=0}^{t} q_i(v) \cdot ((1 + \varepsilon)^{t+1} - (1 + \varepsilon)^i) > f_v$.)

In Algorithm 4, step 2 is just local computation, so we focus on Step 1 which requires the solution to the problem of computing neighborhood sizes.

Cohen's algorithm starts by assigning to each vertex $v$ a rank $\text{rank}(v)$ chosen uniformly from $[0, 1]$. These ranks induce a random permutation of the vertices. To compute the size estimate of a neighborhood, say $B(v, d)$, for a vertex $v$ and real $d > 0$, Cohen's algorithm finds the smallest rank of a vertex in $B(v, d)$. It is then shown (in Section 6, [11]) that the expected value of the smallest rank in $B(v, d)$ is $1/(1 + |B(v, d)|)$. Thus, in expectation, the reciprocal of the smallest rank in $B(v, d)$ is (almost) identical to $|B(v, d)|$. To obtain a good estimate of $|B(v, d)|$ with high probability, Cohen simply repeats the above-described procedure independently a bunch of times and shows the following concentration result on the average estimator.
Theorem 21. (Cohen [11]) Let \( v \) be a vertex and \( d > 0 \) a real. For \( 1 \leq i \leq \ell \), let \( R_i \) denote the smallest rank of a vertex in \( B(v, d) \) obtained in the \( i^{th} \) repetition of Cohen’s neighborhood-size estimation procedure. Let \( \bar{R} \) be the average of \( R_1, R_2, \ldots, R_\ell \). Let \( \mu = 1/(1 + |B(v, d)|) \). Then, for any \( 0 < \varepsilon < 1 \),

\[
\Pr(|\bar{R} - \mu| \geq \varepsilon \mu) = \exp(-\Omega(\varepsilon^2 \cdot \ell)).
\]

This theorem implies that \( \ell = O(\log n/\varepsilon^2) \) repetitions suffice for obtaining \((1 \pm \varepsilon)\)-factor estimates w.h.p. of the sizes of \( B(v, d) \) for all \( v \) and all \( d \).

In [4], the authors show that Algorithm 5 can simulate Cohen’s neighborhood size estimation framework in the \( k \)-machine model in \( \tilde{O}(n/k) \) rounds.

Algorithm 5: NbdSizeEstimates\((G, \varepsilon)\)

1. \( \varepsilon' := \varepsilon/(\varepsilon + 4) \); \( t := [2 \log_{1+\varepsilon'} n] \); \( \ell := \lceil c \log n/(\varepsilon')^2 \rceil \)
2. for \( j := 1, \ldots, \ell \) do
   3. Local Computation. Each machine \( m_j \) picks a rank \( \text{rank}(v) \) for each vertex \( v \in H(m_j) \), chosen uniformly at random from \([0, 1]\). Machine \( m_j \) then rounds \( \text{rank}(v) \) down to the closest \((1 + \varepsilon')^i/n^2\) for integer \( i \geq 0 \)
4. for \( i := 0, 1, \ldots, t - 1 \) do
5. \( T_i := \{ v \in V \mid \text{rank}(v) = (1 + \varepsilon')^i/n^2 \} \)
6. Compute a \((1 + \varepsilon)\)-approximate solution to MSSP using \( T_i \) as the set of sources; let \( \tilde{d}(v, T_i) \) denote the computed approximate distances
7. Local Computation. Machine \( m_j \) stores \( \tilde{d}(v, T_i) \) for each \( v \in H(m_j) \)
8. end
9. end

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

Lemma 22. 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 \( \frac{1}{1+\varepsilon} \tilde{r}_v \leq \bar{r}_v \leq (1 + \varepsilon)\tilde{r}_v \) where \( r_v \) is the actual radius of \( v \) satisfying \( f_v = \sum_{u \in B(v, r_v)} (r_v - \tilde{d}(v, u)) \).

3.1.2 Greedy Phase

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 the 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 \((\varepsilon, d)\)-MIS in [4].

Definition 23 ((\(\varepsilon, d)\)-approximate MIS). For an edge-weighted graph \( G = (V, E) \), and parameters \( d, \varepsilon > 0 \), an \((\varepsilon, d)\)-approximate MIS is a subset \( I \subseteq V \) such that

1. For all distinct vertices \( u, v \in I \), \( d(u, v) \geq \frac{d}{1+\varepsilon} \).
There are \( \approx \) approximate solution rounds.

Theorem 25.

Lemma 24.

Each machine \( m_j \) initializes \( U_j := \emptyset \)

/* Let \( W_j \) denote \( W \cap H(m_j) \).

2 for \( i = 0, 1, \ldots, \lfloor \log n \rfloor \) do

1 for \( [c \log n] \) iterations do

Each machine \( m_j \) marks each vertex \( v \in W_j \) with probability \( 2^i / n \)

/* Let \( R_j \subset W_j \) denote the set of marked vertices hosted by \( m_j \),

4 let \( R := \bigcup_{j=1}^{k} R_j \)

5 Solve an instance of the EXCLUSIVEMSSP problem using \( R \) as the set of sources (see Lemma 20) to obtain \( (1 + \varepsilon) \)-approximate distances \( \tilde{d} \)

6 Each machine \( m_j \) computes \( T_j := \{ v \in R_j \mid d(v, R \setminus \{ v \}) > d \} \)

7 Each Machine \( m_j \) sets \( U_j := U_j \cup T_j \)

8 Solve an instance of the MSSP problem using \( T \) as the set of sources (see Lemma 19) to obtain \( (1 + \varepsilon) \)-approximate distances \( \tilde{d} \)

9 Each machine \( m_j \) computes \( Q_j = \{ v \in W_j \mid d(v, T) \leq (1 + \varepsilon)d \} \)

10 Each machine \( m_j \) sets \( W_j := W_j \setminus (T_j \cup Q_j) \)

11 end

12 end

13 return \( U := \bigcup_{j=1}^{k} U_j \)

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 in the \( k \)-machine model which we describe in Algorithm [5].

Lemma 24. Algorithm [6] finds 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 [6].

Our \( k \)-machine model implementation of the Robust FACLoc algorithm is summarized in Algorithm [7]. 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 next section, and as a result we get the following theorem.

Theorem 25. In \( \tilde{O}(\text{poly}(1/\varepsilon) \cdot n/k) \) rounds, whp, Algorithm [7] finds a factor \( 5 + O(\varepsilon) \) approximate solution \((C', F')\) to the Robust FACLoc problem for any constant \( \varepsilon > 0 \).

Proof. There are \( O(\log(1+\varepsilon) \frac{1}{(1+\varepsilon)}) = O(\log n) \) iterations of the outer for loop, where a facility with the highest opening cost from the range \([(1+\varepsilon)^t, (1+\varepsilon)^{t+1})\]. The guess can be broadcast to all the machines, and they can modify their part of the instance appropriately (without actually removing the facilities from the metric graph). This extra factor is absorbed by the tilde notation, provided that each iteration of the for loop takes \( \tilde{O}(n/k) \) rounds. We can also estimate the cost of a solution within a factor of \( (1 + O(\varepsilon)) \) factor in \( \tilde{O}(n/k) \) rounds – the details can be found in [4]. Since there are \( O(\log n) \) candidate solutions to find a minimum-cost solution from, in line [24] this step can also be implemented in \( \tilde{O}(n/k) \) rounds.
Algorithm 7: \textsc{RobustFacLocDist}(F, C, p)

/* Recall \( \ell := |C| - p \) */

\begin{algorithmic}
\FOR {\( t = 1, \ldots, O(\log n) \)}
\STATE Let \( i_e \in F \) be a most expensive facility from the facilities with opening costs in the range \([(1 + \varepsilon)^t, (1 + \varepsilon)^{t+1})\)
\STATE Modify the facility opening costs to be
\[ f_i' = \begin{cases} 
+\infty & \text{if } f_i > f_{i_e} \\
0 & \text{if } i = i_e \\
f_i & \text{otherwise}
\end{cases} \]
\STATE /* Radius Computation Phase: */
\STATE Call the \textsc{RadiusComputation} algorithm (Algorithm 4) to compute approximate radii.
\STATE /* Greedy Phase: */
\STATE \( F' = \emptyset, \quad C' = \emptyset, \quad O' = C \)
\FOR {\( i = 0, 1, 2, \ldots \)}
\STATE Let \( W \) be the set of vertices \( w \in F \) across all machines with \( \tilde{r}_w = \tilde{r} = (1 + \varepsilon)^i \)
\STATE Using Lemma 19 remove all vertices from \( W \) within approximate distance \( 2(1 + \varepsilon)^3 \cdot \tilde{r} \) from \( F' \)
\STATE \( I \leftarrow \text{APPROXIMATEMIS}(G, W, 2(1 + \varepsilon)^3 \cdot \tilde{r}, \varepsilon) \)
\STATE \( F' \leftarrow F' \cup I \)
\STATE Using Lemma 19 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 \( i \)
\STATE \textbf{if} \( |O'| \leq \ell \) \textbf{then} \textbf{break}
\ENDFOR
\STATE /* Outlier Determination Phase: */
\STATE \textbf{if} \( |O'| > \ell \) \textbf{then}
\STATE Using Lemma 19 find \( O_1 \subseteq O' \), a set of \( |O'| - \ell \) clients that are closest to facilities in \( F' \).
\STATE \( C' \leftarrow C' \cup O_1, \quad O' \leftarrow O' \setminus O_1. \)
\STATE \textbf{end}
\STATE \textbf{else if} \( |O'| < \ell \) \textbf{then}
\STATE Using Lemma 19 find \( O_2 \subseteq C \setminus O' \), a set of \((\ell - |O'|)\) clients that are farthest away from facilities in \( F' \)
\STATE \( C' \leftarrow C' \setminus O_2, \quad O' \leftarrow O' \cup O_2. \)
\STATE \textbf{end}
\STATE \textbf{end}
\STATE \textbf{if} \( |O'| \leq \ell \) \textbf{then}
\STATE \textbf{return} \( (C'_t, F'_t) \) with a minimum cost
\STATE \textbf{end}
\ENDFOR
\end{algorithmic}
Each iteration of the for loop \(\text{Algorithm 7}\) consists of two phases namely, the Radius Computation and Greedy Phases. We bound the running time of both these phases separately. By Lemma \(22\) we know that the radius computation phase of Algorithm \(\text{Algorithm 7}\) requires \(\tilde{O}(n/k)\) rounds. In the for loop on line \(6\) there are at most \(O(\log_{1+\epsilon} nN) = O(\log nN) = O(\log n)\) possible values of \(i\) and hence at most \(O(\log n)\) iterations (where \(N = \text{poly}(n)\) is the largest edge weight).

Each individual step in the greedy phase of Algorithm \(\text{Algorithm 7}\) takes \(\tilde{O}(n/k)\) rounds therefore we conclude that the overall running time is \(\tilde{O}(n/k)\) rounds. The proof of the approximation guarantee appears in the next section.

### 3.1.3 Analysis of the Algorithm

Similar to the sequential algorithm analysis, we analyze the cost of the corresponding costlier solution \((C', F')\). In order to alleviate excessive notation, we will henceforth refer to the solution \((C', F')\) as \((C', F')\) and \(\mathcal{O}'\) as \(O'\). We now restate the standard primal and dual for the Robust Facility Location problem.

Let \(r_i\) be the radius value of \(i\) satisfying \(f_i = \sum_{j \in B(i, r_i)} (r_i - c_{ij})\) and let \(\tilde{r}_i\) be the approximate radius value of \(i\) computed during Algorithm \(\text{Algorithm 7}\).

First, we construct a feasible dual solution \((v, w, q)\). For a facility \(i \in F\) and client \(j \in C\), let \(w_{ij} := \max\{0, r_i - c_{ij}\}\). Let \(q := \max_{j \in C'} v_j'/(1 + \epsilon)^4\) (recall that \(v_j' := \min_{i \in F} \max\{r_i, c_{ij}\} = \min_{i \in F} c_{ij} + w_{ij}\)). Now, for a client \(j \in C\), define \(v_j\) as follows:

\[
    v_j = \begin{cases} 
        v_j'/(1 + \epsilon)^4 & \text{if } j \in C' \\
        q & \text{if } j \in O' 
    \end{cases}
\]

> **Claim 26.** If a client \(j \in C''\) then \(v_j' \leq (1 + \epsilon)^2 \tilde{r}_i\) and if a client \(j \in O''\) then \(v_j' \geq (1 + \epsilon)^{-2} \tilde{r}_i\) where \(i\) is the last iteration of the for loop (lines \(6-13\)).

**Proof.** If \(j \in C''\) then it must be added to \(C_i\) for some iteration \(i' \leq i\). Let us assume wlog that \(j\) was added to \(C_i\). Therefore, there must be some \(i' \in F_i\) such that \(j \in B(i', (1 + \epsilon)\tilde{r}_i)\). This means that \(c_{ij} \leq (1 + \epsilon)\tilde{r}_i\)

\[
v_j' = \min_{i' \in F_i} \max\{r_{i'}, c_{ij}\} \leq \max\{r_{i'}, c_{ij}\}
\]

\[
\leq (1 + \epsilon)^2 \tilde{r}_i
\]

Since we compute approximate shortest paths, if client \(j\) is not added to any \(C_i\) then for all \(i' \in F_i, j \notin B(i', \tilde{r}_i)\) (otherwise we would add \(j\) to \(C_i\)). Therefore if \(j \in O''\), for all \(i' \in F_i\), \(c_{ij} > \tilde{r}_i\). So we have,

\[
v_j' = \min_{i' \in F_i} \max\{r_{i'}, c_{ij}\} \geq (1 + \epsilon)^{-2} \min_{i' \in F_i} \max\{r_{i'}, c_{ij}\}
\]

Because for facilities \(i'' \in F_i, r_{i''} \geq (1 + \epsilon)^{-2} \tilde{r}_i\) \(r_{i''} \geq (1 + \epsilon)^{-2} \tilde{r}_i\) as we process facilities in increasing order of \(\tilde{r}\). Therefore,

\[
v_j' \geq (1 + \epsilon)^{-2} \min_{i' \in F_i} \max\{r_{i'}, \tilde{r}_i\}
\]

\[
\geq (1 + \epsilon)^{-2} \min_{i' \in F_i} \max\{(1 + \epsilon)^{-2} \tilde{r}_{i'}, \tilde{r}_i\}
\]

\[
= (1 + \epsilon)^{-2} \tilde{r}_i
\]
Lemma 27. The solution $(v, w, q)$ is a feasible solution to the dual LP relaxation. 

Proof. Note that constraints 6, 7, and 8 of the dual are satisfied by construction and so is constraint 5 for clients $j \in C'$. Therefore, in order to show that the solution $(v, w, q)$ is feasible, we have to show that constraint 5 is satisfied for all clients $j \in O'$. To this end, we consider the following two cases.

Case 1. We enter the outlier determination phase after iterating over all facilities in $F$. Therefore, we have shown that $(v, w, q)$ is a feasible solution to the dual LP relaxation. We put in $O_1$ the clients from $O''$ with smallest $v_j'$-value. This means that for all $j \in O'$:

$$v_j' \geq \max_{j \in O_1} v_j'$$

$$= \max \left\{ \max_{j \in C'} \frac{v_j'}{(1 + \varepsilon)^2}, \max_{j \in O_1} v_j' \right\} \quad (\max_{j \in C'} v_j' \leq \max_{i \in F} (1 + \varepsilon)^2 \tilde{r}_i \text{ by Claim 26})$$

$$\geq \frac{1}{(1 + \varepsilon)^4} \max \left\{ \max_{j \in C'} v_j', \max_{j \in O_1} v_j' \right\}$$

$$\geq \frac{1}{(1 + \varepsilon)^4} \max_{j \in C'} v_j'$$

$$\geq q$$

Therefore, we conclude that for any $j \in O'$ and $i \in F$, $c_{ij} + w_{ij} \geq v_j' \geq q = v_j$.

Case 2. We enter the outlier determination phase because of the break statement on line 12. Here, $|O''| < \ell$ and $C' \subseteq C''$.

Let $i^*$ be the last iteration of the for loop. Therefore $F_{i^*}$ is the set of facilities we consider in the for loop and $\max_{i \in F_{i^*}} r_i \leq (1 + \varepsilon)^2 \tilde{r}_{i^*}$. We show that $q \leq (1 + \varepsilon)^2 \tilde{r}_{i^*}$.

Recall that by the case assumption we have $|O''| \leq \ell$ and hence $C' \subseteq C''$. All clients $j \in C''$ were part of $C_i$ for some $i \in F_{i^*}$ and by Claim 26 we have $v_j' \leq (1 + \varepsilon)^2 \tilde{r}_i \leq (1 + \varepsilon)^2 \tilde{r}_{i^*}$. Therefore,

$$q = \max_{j \in C'} v_j' \leq \max_{j \in C''} v_j' \leq \max_{i \in F_{i^*}} (1 + \varepsilon)^2 \tilde{r}_i \leq (1 + \varepsilon)^2 \tilde{r}_{i^*}$$

Let $j \in O'$ be an outlier client. If $j \in O''$, then for any facility $i \in F$,

$$c_{ij} + w_{ij} \geq v_j'$$

$$\geq (1 + \varepsilon)^{-2} \tilde{r}_i$$

$$\geq q \quad \text{(by Claim 26)}$$

otherwise, $j \in O_2$ and was added to $O'$ because it had highest $v_j'$ value among $C_{i^*}$. Therefore, by Claim 26 it follows that for any facility $i \in F$, $c_{ij} + w_{ij} \geq v_j' \geq \max_{j \in C'} v_j'(1 + \varepsilon)^{-4} = q = v_j$

From the case analysis it follows that $v_j \leq c_{ij} + w_{ij}$ for all $j \in O'$ and for all $i \in F$. Therefore, we have shown that $(v, w, q)$ is a dual feasible solution. △
For the approximation guarantee we can now focus just on the clients in \( C' \) because the only contribution that the clients in \( O' \) make to the dual objective function is to cancel out the \(-\rho_i\) term and hence they do not affect the approximation guarantee. We call a facility \( i \) the bottleneck of \( j \) if \( v_j' = \max\{r_i, c_{ij}\} = c_{ij} + w_{ij} \).

Throughout this section, we condition on the event that the outcome of all the randomized algorithms is as expected (i.e., the “bad” events do not happen). Note that this happens with w.h.p. We first need the following facts along the lines of \( 39 \).

Lemma 28 (Modified From Lemma 8 Of \[39\]). There exists a total ordering \( \prec \) on the facilities in \( F \) such that \( u \prec v \implies \tilde{r}_u \leq \tilde{r}_v \), and \( v \) is added to \( F' \) if and only if there is no previous \( u \prec v \) in \( F' \) such that \( c_{uv} \leq 2(1+\varepsilon)^2\tilde{r}_v \).

Proof Sketch. The ordering is obtained by enumerating the facilities processed in each iteration (with arbitrary order given to facilities in the same iteration). The facilities in \( I \) that are included in \( F' \) before the rest of the vertices of \( W \). The lemma follows because of line \( 8 \) (if \( u \) and \( v \) are processed in different iterations) the definition of \((\varepsilon, d)\)-approximate MIS (if \( u \) and \( v \) are processed in the same iteration).

Claim 29 (Modified From Claim 9.2 Of \[39\]). For any two distinct vertices \( u, v \in F' \), we have that \( c_{uv} > 2(1+\varepsilon)^2 \cdot \max\{\tilde{r}_u, \tilde{r}_v\} \).

Proof Sketch. Without loss of generality, assume that \( u \prec v \), so \( \tilde{r}_u \leq \tilde{r}_v \). From Lemma 28 we have \( c_{uv} > 2(1+\varepsilon)^2\tilde{r}_v \).

We now prove a few claims about the dual solution.

Claim 30. For any \( i \in F \) and \( j \in C \), \( \bar{r}_i \leq (1+\varepsilon)^2(c_{ij} + w_{ij}) \). Furthermore if for some \( i \in F \) and \( j \in C \), \( w_{ij} > 0 \), then \( \bar{r}_i \geq (1+\varepsilon)^{-2}(c_{ij} + w_{ij}) \).

Proof. We have \( w_{ij} = \max\{0, r_i - c_{ij}\} \geq r_i - c_{ij} \) and therefore \( \bar{r}_i \leq (1+\varepsilon)^2r_i \leq (1+\varepsilon)^2(c_{ij} + w_{ij}) \).

If for some \( i \in F \) and \( j \in C \), \( w_{ij} > 0 \), then \( w_{ij} = r_i - c_{ij} \) which means that \( \bar{r}_i \geq (1+\varepsilon)^{-2}r_i = (1+\varepsilon)^{-2}(c_{ij} + w_{ij}) \).

Claim 31. If \( i \) is a bottleneck for \( j \in C' \), then \( (1+\varepsilon)^2v_j \geq \bar{r}_i \).

Proof. For \( j \in C' \), we have
\[
(1+\varepsilon)^2v_j = (1+\varepsilon)^{-2}v_j' = (1+\varepsilon)^{-2}\max\{c_{ij}, r_i\} \geq (1+\varepsilon)^{-2}r_i \geq \bar{r}_i
\]

Claim 32. If \( i \in F' \) is a bottleneck for \( j \in C' \), then \( w_{i'j} = 0 \) for all \( i' \in F' \), where \( i' \neq i \).

Proof. Assume for contradiction that \( w_{i'j} > 0 \), i.e., \( r_{i'} \geq c_{i'j} \) for some \( i' \in F' \), \( i' \neq i \).

If \( r_i \geq c_{ij} \), then \( v_j' = r_i \leq \max\{c_{ij}, r_i\} = r_i \). In this case, \( c_{i'j} \leq c_{ij} + c_{ij} \leq 2r_i \).

Otherwise, if \( c_{ij} > r_i \), then \( v_j' = c_{ij} \leq \max\{c_{ij}, r_i\} = r_i \). Here too we have, \( c_{i'j} \leq c_{ij} + c_{ij} \leq 2r_i \).

In either case, \( c_{i'j} \leq 2r_i \leq 2\max\{r_i, r_i\} \leq (1+\varepsilon)^2\max\{\bar{r}_i, \bar{r}_i\} \), which is a contradiction to Claim 29 since at most one of \( i', i \) can be added to \( F' \).
Claim 33. If a closed facility \( i \notin F' \) is the bottleneck for \( j \in C' \), and if an open facility \( i' \in F' \) caused \( i \) to close, then \( c_{i'j} \leq 3(1 + \epsilon)^8 \cdot v_j \).

Proof. 

\[ c_{i'j} \leq c_{ij} + c_{ij} \]  \hspace{1cm} \text{(Triangle inequality)}
\[ \leq 2(1 + \epsilon)^2 \tilde{r}_i + c_{ij} \]  \hspace{1cm} \text{\( (i' \text{ caused } i \text{ to close, so using Lemma 28}) \)}
\[ \leq 2(1 + \epsilon)^2 \cdot (1 + \epsilon)^2 (w_{ij} + c_{ij}) + c_{ij} \]  \hspace{1cm} \text{(Using Claim 31)}
\[ \leq 2(1 + \epsilon)^4 w_{ij} + (2(1 + \epsilon)^4 + 1) \cdot c_{ij} \]
\[ \leq \max\{2(1 + \epsilon)^4, 2(1 + \epsilon)^4 + 1\} \cdot v_j' \]  \hspace{1cm} \text{(Since } \epsilon \text{ is the bottleneck for } j \text{)}
\[ \leq 3(1 + \epsilon)^8 \cdot v_j \]  \hspace{1cm} \text{(By definition of } v_j \text{)}

Now we state the main lemma that uses the dual variables for analyzing the cost.

Lemma 34. For any \( i \in C' \), there is some \( i' \in F' \) such that \( c_{ij} + w_{ij} \leq 3(1 + \epsilon)^8 v_j \).

Proof. Fix a client \( j \in C' \), and let \( \epsilon \) be its bottleneck. We consider different cases. Here, \( c_{ij} \) should be seen as the connection cost of \( j \), and \( w_{ij} \), the cost towards opening of a facility (if \( w_{ij} > 0 \)).

Case 1: \( i \in F' \).

In this case, by Claim 33, \( w_{ij} = 0 \) for all other \( i' \in F' \).
If \( c_{ij} < r_i \), then \( w_{ij} > 0 \), and \( v_j' = c_{ij} + w_{ij} \). Therefore, \( (1 + \epsilon)^4 v_j = v_j' \) pays for the connection cost as well as towards the opening cost of \( i \).
Otherwise, if \( c_{ij} \geq r_i \), then \( w_{ij} = 0 \). Also, \( w_{ij}' = 0 \) for all other \( i' \in F' \). Therefore, \( j \) does not contribute towards opening of any facility in \( F' \). Also, we have \( v_j' = \max\{c_{ij}, r_i\} = c_{ij}, \) i.e., \( (1 + \epsilon)^4 v_j \) pays for \( j \)'s connection cost.

Case 2: \( i \notin F' \) and \( w_{ij} = 0 \) for all \( i \in F' \).

Let \( i' \in F' \) be the facility that caused \( i \) to close. From Claim 33, we have \( c_{i'j} \leq 3(1 + \epsilon)^8 v_j \), i.e., \( 3(1 + \epsilon)^8 v_j \) pays for the connection cost of \( j \).

Case 3: \( i \notin F' \), and there is some \( i' \in F' \) with \( w_{ij} > 0 \). But \( i' \) did not cause \( i \) to close.
Since \( w_{ij} > 0 \), by Claim 33 \( \tilde{r}_{i'} \geq (1 + \epsilon)^{-2} (c_{ij} + w_{ij}) \).
Let \( i \) be the facility that caused \( i \) to close. Therefore, by Claim 33 we have \( c_{ij} \leq 3(1 + \epsilon)^8 v_j \).
Now, since \( i \) and \( i' \) both belong to \( F' \), by Claim 29 \( c_{ij} > (1 + \epsilon)^2 \max\{\tilde{r}_{i}, \tilde{r}_{i'}\} \geq 2\tilde{r}_{i'} \geq 2(c_{ij} + w_{ij}) \).
Therefore, by triangle inequality, \( c_{ij} + c_{ij} \geq c_{i'j} > 2(c_{ij} + w_{ij}) \) which implies \( c_{ij} > c_{i'j} + 2w_{ij} \).
It follows that, \( c_{ij} + w_{ij} \leq c_{ij} + 2w_{ij} < c_{ij} \leq 3(1 + \epsilon)^8 v_j \). Therefore, \( 3(1 + \epsilon)^8 v_j \) pays for the connection cost of \( j \) to \( i' \), and its contribution towards opening of \( i' \).

Case 4: \( i \notin F' \), but for \( i' \in F' \) that caused \( i \) to close has \( w_{ij} > 0 \).
Again, since \( w_{ij} > 0 \), by Claim 33 \( \tilde{r}_{i'} \geq (1 + \epsilon)^{-2} (c_{ij} + w_{ij}) \).
From Claim 33, we have that \( (1 + \epsilon)^2 w_{ij} \geq \tilde{r}_{i} \). Then, since \( i' \) caused \( i \) to close, we have \( \tilde{r}_{i} \geq \tilde{r}_{i'} \geq (1 + \epsilon)^{-2} (c_{ij} + w_{ij}) \). This implies \( (1 + \epsilon)^4 v_j \geq c_{ij} + w_{ij} \), i.e., \( (1 + \epsilon)^4 v_j \) pays for the connection cost of \( j \) to \( i' \), and its contribution towards opening of \( i' \).

Now we are ready to prove the approximation guarantee of the algorithm.
Theorem 35. \( \text{cost}_e(C', F') \leq 3(1 + \varepsilon)^8 \cdot \text{cost}_e(C^*_e, F^*_e) + f_{i_e} \)

Proof. Recall that \( f_{i_e} \) denotes the cost of the most expensive facility in an optimal solution. Furthermore, notice that for any facility \( i \in F' \setminus \{i^*\} \), the clients in the ball \( B(i, r_i) \subseteq C' \). However, if \( i^* \in F' \), some of the clients in \( B(i^*, r_{i^*}) \) may have been removed in the outlier determination phase, and therefore it may not get paid completely by the dual variables \( v_j \).

Therefore,

\[
\text{cost}_e(C', F') = \sum_{j \in C'} d(j, F) + \sum_{i \in F' \setminus \{i^*\}} f_i + f_{i^*} \\
\leq 3(1 + \varepsilon)^8 \cdot \sum_{j \in C'} v_j + f_{i^*} \quad \text{(From Lemma 34)} \\
= 3(1 + \varepsilon)^8 \cdot \left( \sum_{j \in C'} v_j - q \ell \right) + f_{i^*} \quad \text{(For } j \in O', v_j = q, \text{ and } |O'| = \ell) \\
\leq 3(1 + \varepsilon)^8 \cdot \left( \sum_{j \in C'} v_j - q \ell \right) + f_{i_e} \quad \text{(Since } i_e \text{ is the most expensive facility)}
\]

Since \( (v, w, q) \) is a feasible dual solution, its cost is a lower bound on the cost of any integral optimal solution. Therefore, the theorem follows.

Therefore, we can apply corollary 2 with \( \alpha = 1 + \varepsilon, \beta = 3(1 + \varepsilon)^8, \gamma = 1 \) to get the following approximation guarantee –

Theorem 36. The solution returned by Algorithm 7 is a \( 5 + O(\varepsilon) \) approximation to the Robust Facility Location problem.

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 25, we get the following result.

Theorem 37. 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 7 reduces the task of finding an approximate solution to the Robust FACLOC problem in the implicit metric setting to \( \text{poly log } n \) 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 [6].
Theorem 38. Let $A$ be a $T$ round BCC algorithm that uses $\tilde{O}(n)$ local memory at each node. One can simulate $A$ 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 39. 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 [23], which presents an $O(1)$-approximation algorithm to FACLOC running in $O(\log \log n)$ rounds whp. The algorithms in [21] and in [23] 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 [23] 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, 23], 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, 23], 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 [23] 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 40. 2-ruling sets for all graphs $G_i$, $1 \leq i \leq c$, can be computed in $O(\log \log \log n)$ rounds whp.

Proof. The proof is simply an accounting of the communication that occurs in each phase of the 2-ruling set algorithm in [23]. The accounting establishes that there is enough bandwidth in the Congested Clique model to allow for $c$ instances of the algorithm (one for each graph $G_i$) to run in parallel, without increasing the number of rounds by more than a constant-factor. The 2-ruling set algorithm of [23] consists of 5 phases (in this order): (1) Lazy Degree Decomposition phase, (2) Speedy Degree Decomposition phase, (3) Vertex Selection phase, (4) High Degree Vertex Removal phase, (5) MIS in Low-Degree Graphs phase. Phases (1)-(4) are described in detail in [23], whereas Phase 5 is described in [18].
In the Lazy Degree Decomposition phase and the Vertex Selection phases, each vertex communicates by broadcasting a bit. To run \( c \) instances of these two phases, a vertex can simply package the \( c \) bits it needs to send (one for each instance) into \( O(1) \) messages of size \( O(\log n) \) each and use \( O(1) \) rounds to perform the communication.

The key part of the Speedy Degree Decomposition phase is for each vertex \( v \in V \) to learn \( B_{G_t}(v, \lfloor \log \log n \rfloor) \), the topology up to distance \( \lfloor \log \log n \rfloor \) hops of the graph \( G_t \) that is active after the Lazy Degree Decomposition phase. This phase consists of \( \lfloor \log \log \log n \rfloor \) iterations (each of which can be implemented in \( O(1) \) rounds) and in iteration \( i, 0 \leq i \leq \lfloor \log \log \log n \rfloor - 1 \), vertex \( v \)'s knowledge expands from \( B_{G_i}(v, 2^i) \) to \( B_{G_{i+1}}(v, 2^{i+1}) \). This is achieved by each vertex \( v \) sending \( B_{G_{i}}(v, 2^i) \) to all vertices in \( B_{G_{i}}(v, 2^i) \) in iteration \( i \). If \( G_t \) has maximum degree \( \Delta \), then \( B_{G_{i}}(v, 2^i) \) contains \( O(\Delta^{2^i}) \) vertices and \( O(\Delta^{2^i+1}) \) edges. Thus each vertex \( v \) needs to send (and receive) a total of \( O(\Delta^{2^i+1+2^i}) = O(\Delta^{3\log \log n}) \) messages.

If the Lazy Degree Decomposition phase is run for \( t \) rounds, then \( \Delta \leq n^{1/2^t} \). Currently, the Lazy Degree Decomposition is run for \( t = 1 + \lfloor \log \log \log n \rfloor \) iterations. If we run it for two additional iterations, then \( \Delta \leq n^{1/8 \log \log n} \) and therefore the total number of messages a vertex needs to send (receive) per round in an instance of the Speedy Degree Decomposition phase is \( O(n^{1/8}) \). So even if we were to run \( c = O(\log n) \) instances of this phase in parallel, the number of messages a vertex needs to send (receive) per round is \( O(n) \). Therefore, Lenzen’s routing protocol can be used to get all of these messages to their destination in \( O(1) \) rounds and therefore all \( c \) instances of the algorithm can complete an iteration of the Speedy Degree Decomposition phase in \( O(1) \) rounds.

The High Degree Vertex Removal phases starts with a set \( S \) of vertices that are active after the Vertex Selection phase. A leader vertex (e.g., a vertex with lowest ID) generates a random ranking (permutation) of the vertices in \( S \). Let the parameter \( \delta = \log^2 n \). The subgraph induced by \( P \subseteq S \), the set of vertices with rank in \([1 \ldots |S|/\delta] \), is sent to the leader. To run \( c \) instances of this phase, we simply use \( c \) distinct leaders (e.g., the \( c \) vertices with lowest IDs), one for each instance of the algorithm. This permits the random rank generation for the \( c \) instances to complete in parallel in \( O(1) \) rounds. In the proof of Theorem 2.15 in [18], it is shown that the number of edges of the graph induced by \( P \), incident on a vertex is \( O(n/\log^2 n) \) whp. It is also shown that whp the total number of edges in this induced subgraph is \( O(n) \). Therefore, each vertex needs to send \( O(n/\log^2 n) \) messages to the leader. Even with \( c = O(\log n) \) instances of the algorithm, each vertex needs to send \( O(n/\log n) \) messages. Furthermore, since we are using \( c \) distinct leaders to receive the graphs each of size \( O(n) \), we can use Lenzen’s routing protocol to complete this communication in \( O(1) \) rounds.

Finally, we examine the MIS in Low-Degree Graphs phase. This phase consists of two parts, the first being the gathering by each vertex \( v \) of its \( O(\log \log n) \)-hop neighborhood (see Lemma 2.15 in [18]). The maximum degree of the graph on which we run this phase is \( O(\log^3 n) \) and therefore the accounting that we did for the Speedy Degree Decomposition phase applies. Recall that in the analysis of the Speedy Degree Decomposition phase the maximum degree was bounded above by \( n^{1/8 \log \log n} \). Finally, in the second part of the MIS in Low-Degree Graphs phase, the graph that is still active is gathered and processed at a single leader vertex. It is shown in Lemma 2.11 in [18] that this graph has \( O(n) \) edges and therefore can gathered at the leader in \( O(1) \) rounds. To run \( c = O(\log n) \) instances of this phase, as before, we simply pick \( c \) leaders. Thus there is still enough bandwidth from receiver’s perspective. Also note that there is enough bandwidth from the sender’s perspective because the maximum degree of the graph that enters the MIS in Low-Degree Graphs phase is \( O(\log^3 n) \).
The theorem above and the discussion preceding it leads to the following theorem.

\textbf{Theorem 41.} There is an $O(1)$-approximation algorithm in the Congested Clique model for Robust FACLOC, running in $O(\log \log \log n)$ rounds whp.

\section{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.

\textbf{Theorem 42.} 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.

\section{5 Facility Location with Penalties: Implicit Metric}

\subsection{5.1 The $k$-Machine Algorithm}

In this section we describe how to implement Algorithm 3 in the $k$-machine model. Since the radius computation phase for FACLOC with Penalties is different from the one for Robust FACLOC (Algorithm 4), we first show how to modify Algorithm 4 in order to compute approximate radii for the Penalty version.

\subsubsection{5.1.1 Radius Computation}

In FACLOC with Penalties, the definition of radii differs from that in Robust FACLOC (or the standard Facility Location algorithm) due to the penalties of the clients. In particular, for a vertex $v$, the radius is defined as $r_v$ satisfying the following equation: $f_v = \sum_{u \in V} \max\{\min\{r_u - d(u,v), p_u - d(u,v)\}, 0\}$. Throughout this section, we assume that such an $r_v$ exists otherwise, it can be shown that excluding $v$ as a candidate facility does not affect the cost of any solution. We now show how to appropriately modify the neighborhood computation and the radius computation subroutines.

The key idea is to divide vertices into $O(\log n)$ classes, such that the penalties of the vertices belonging to a particular class are within $1 + O(\varepsilon)$ factor of each other. Then, for any vertex $v$, and for each penalty class, we estimate the number of vertices from that penalty class, in $(1 + \varepsilon) \cdot$-neighborhood of $v$. Once we have these estimates for each range of neighborhoods, they can be used for computation of approximate computation of radii. We formalize this in the following.

First, we assume that we have normalized $f_i, c_{ij}, p_j$, such that any positive quantity is at least 1. Note that we can normalize any given input in this manner in $O(1)$ rounds of the $k$-machine model. Let $P_0 := \{j \in V \mid p_j = 0\}$, and for any integer $t \geq 1$, let $P_t := \{j \in V \mid (1 + \varepsilon)^t \leq p_j < (1 + \varepsilon)^{t+1}\}$. By assumption, the penalties are polynomially bounded in $n$, and hence the total number of penalty classes is $O(\log n)$.

Let $\text{NBDSizeEstimates}(G, \varepsilon, b)$ be a modified version (of the original algorithm, Algorithm 3 in [4], see Algorithm 8) that takes an additional parameter $b \geq 0$, wherein only the vertices in $P_b$ participate. That is, random ranks (as in the original version) are chosen
only for the vertices in $P_b$. However, the neighborhood size estimates are computed for all vertices. The details are straightforward, and are therefore omitted.

| Algorithm 8: NbdSizeEstimates($G, \varepsilon, b$) |
|--------------------------------------------------|
| $\varepsilon' := \varepsilon/(\varepsilon + 4)$; $t = \lceil 2\log_{1+\varepsilon} n \rceil$; $\ell := \lceil c\log n/(\varepsilon')^2 \rceil$ |
| for $j := 1, \ldots, t$ do |
|   Local Computation. Each machine $m_j$ picks a rank rank(v), for each vertex $v \in H(m_j) \cap P_b$, chosen uniformly at random from $[0, 1]$. Machine $m_j$ then rounds rank(v) down to the closest $(1 + \varepsilon')^i/n^2$ for integer $i \geq 0$ |
|   for $i := 0, 1, \ldots, t - 1$ do |
|     $T_i := \{ v \in P_b \mid \text{rank}(v) = (1 + \varepsilon')^i/n^2 \}$ |
|     Compute a $(1 + \varepsilon)$-approximate solution to MSSP using $T_i$ as the set of sources; let $d(v, T_i)$ denote the computed approximate distances |
|   Local Computation. Machine $m_j$ stores $d(v, T_i)$ for each $v \in H(m_j)$ |
| end |
| end |

For a vertex $v \in V$, and for parameters $b, r, \geq 0$, let $B(v, r, b) := B(v, r) \cap P_b$. Then, let $Q(v, r, b)$ denote the query “What is the size of $B(v, r) \cap P_b$?” The details of how to answer this query from the output of NbdSizeEstimates are same as in the original version.

**Lemma 43.** For any vertex $v \in V$, for any $r, b \geq 0$, and for any $\varepsilon > 0$, the modified NbdSizeEstimates algorithm satisfies the following properties.

- For the query $Q(v, r/(1 + \varepsilon), b)$, the algorithm returns an output that is at most $|B(v, r) \cap P_b| \cdot (1 + \varepsilon)$.
- For the query $Q(v, r(1 + \varepsilon), b)$, the algorithm returns an output that is at most $|B(v, r) \cap P_b|/(1 + \varepsilon)$.

We define the following quantities with respect to any vertex $v \in V$. Let $\alpha(v, r, b) := \sum_{u \in B(v, r, b)} \max\{\min\{r - d(u, v), p_u - d(u, v)\}, 0\}$, and let $\alpha(v, r) := \sum_{u \in B(v, r, b)} \max\{\min\{r - d(u, v), p_u - d(u, v)\}, 0\}$. It is easy to see that $\alpha(v, r) = \sum_{b \geq 0} \alpha(v, r, b)$. Finally, let $q_{i, b}(v) := |B(v, (1 + \varepsilon)^i, b)|$.

**Lemma 44.** If $t > b$, then $\alpha(v, (1 + \varepsilon)^t, b) \geq \sum_{i=0}^{t-1} q_{i, b} ((1 + \varepsilon)^{i+1} - (1 + \varepsilon)^i)$. Otherwise, $\alpha(v, (1 + \varepsilon)^t, b) \geq \sum_{i=0}^{t-1} q_{i, b} ((1 + \varepsilon)^{i+1} - (1 + \varepsilon)^i) + \sum_{i=b}^{t-1} q_{i, b} ((1 + \varepsilon)^{i+1} - (1 + \varepsilon)^i)$.

**Proof.** First, let $t > b$. And consider,

$$
\begin{align*}
\alpha(v, (1 + \varepsilon)^t, b) &= \sum_{i=0}^{t-1} \alpha(v, (1 + \varepsilon)^{i+1}, b) - \alpha(v, (1 + \varepsilon)^i, b) \\
&\geq \sum_{i=0}^{t-1} \sum_{u \in B(v, (1 + \varepsilon)^i, b)} \left( \min\{(1 + \varepsilon)^{i+1} - d(u, v), (1 + \varepsilon)^b - d(u, v)\} \\
&\quad - \min\{(1 + \varepsilon)^i - d(u, v), (1 + \varepsilon)^{b+1} - d(u, v)\} \right) \\
&= \sum_{i=0}^{t-1} \sum_{u \in B(v, (1 + \varepsilon)^i, b)} (1 + \varepsilon)^{i+1} - d(u, v) - ((1 + \varepsilon)^i - d(u, v))
\end{align*}
$$
Lemma 45. By Lemma 43, we have the following for $\lambda$

$$\geq \alpha(v, (1 + \varepsilon)^i, b) + \sum_{i=0}^{t-1} q_{i,b} ((1 + \varepsilon)^{i+1} - (1 + \varepsilon)^i).$$

Now, if $t \leq b$, then

$$\alpha(v, (1 + \varepsilon)^t, b) = \alpha(v, (1 + \varepsilon)^b, b) + \sum_{i=b}^{t-1} \alpha(v, (1 + \varepsilon)^{i+1}, b) - \alpha(v, (1 + \varepsilon)^i, b)$$

$$\geq \alpha(v, (1 + \varepsilon)^b, b) + \sum_{i=b}^{t-1} q_{i,b} ((1 + \varepsilon)^{i+1} - (1 + \varepsilon)^i).$$

Where final step uses similar arguments from the previous case. Now, we consider,

$$\alpha(v, (1 + \varepsilon)^b, b) = \sum_{u \in B(v, (1 + \varepsilon)^b, b))} (1 + \varepsilon)^b - d(u,v)$$

$$= \sum_{i=0}^{b-1} \sum_{u \in B(v, (1 + \varepsilon)^{i+1}, b) \setminus B(v, (1 + \varepsilon)^i, b))} (1 + \varepsilon)^b - d(u,v)$$

$$\geq \sum_{i=0}^{b-1} q_{i,b}(v) ((1 + \varepsilon)^b - (1 + \varepsilon)^{i+1})$$

For a vertex $v$, and for any $t, b \geq 0$, let $\lambda(v, t, b)$ denote the appropriate lower bound on $\alpha(v, (1 + \varepsilon)^t, b)$, given by Lemma 44 (based on two different cases). Let $\tilde{\lambda}(v, t, b)$ be the quantity obtained by replacing $q_{i,b}(v)$ by the approximate neighborhood estimate $\tilde{q}_{i,b}(v)$ in the lower bound $\lambda(v, t, b)$. We now state the Radius Computation algorithm.

**Algorithm 9: RadiusComputation Algorithm (Version 2)**

1. **Neighborhood-Size Computation.** Call the \textsc{NbdSizeEstimates} algorithm (Algorithm 5) to obtain approximate neighborhood-size estimates $\tilde{q}_{i,b}(v)$ for all integers $i \geq 0, b \geq 0$ and for all vertices $v$.

2. **Local Computation.** Each machine $m_j$ computes $\tilde{r}_v$ locally, for all vertices $v \in H(m_j)$ using the formula $\tilde{r}_v := (1 + \varepsilon)^{t-1}$ where $t \geq 1$ is the smallest integer for which $\sum_{b \geq 0} \tilde{\lambda}(v, t, b) > f_v$. If there is no such integer, define $\tilde{r}_v = \infty$.

We have the following bounds on the approximate radius computed by the algorithm.

**Lemma 45.** For every $v \in V$, \(\frac{r_v}{(1+\varepsilon)^{t-1}}\leq \tilde{r}_v \leq (1+\varepsilon)^2 \cdot r_v\)

**Proof.** By Lemma 44 we have the following for $i \geq 1$:

$$\frac{1}{(1+\varepsilon)^{i}} \cdot q_{i-1,b}(v) \leq \tilde{q}_{i,b}(v) \leq (1+\varepsilon) \cdot q_{i+1,b}(v).$$

Now, let $t$ be the smallest integer for which $\sum_{b \geq 0} \tilde{\lambda}(v, t, b) > f_v$. Now, by Lemma 44 we have that $\lambda(v, t, b) \geq \alpha(v, (1 + \varepsilon)^t, b)$. Now, using arguments very similar to those in the proof of Lemma 8 of [4], one can show the following inequality.
Theorem 46. We state the standard primal and dual linear programming relaxations for facility location with penalties in Figure 2. For

\[ \alpha(v, (1 + \varepsilon)^{t-2}) \leq \sum_{k \geq 0} \lambda(v, t, b) \leq \alpha(v, (1 + \varepsilon)^{t+1}). \]

Recall that \( \alpha(v, r) = \sum_{b \geq 0} \alpha(v, r, b). \)

Therefore, there must exist a value \( r_v \in [(1 + \varepsilon)^{t-3}, (1 + \varepsilon)^{t+1}] \), such that \( \alpha(v, r_v) = f_v. \)

The Lemma follows, because \( \tilde{r}_v = (1 + \varepsilon)^{t-1}. \)

\[
\begin{align*}
\text{Algorithm 10: PENALTYFacLoc}(G, F; C, p) \\
&/* \text{ Radius Computation Phase: } */ \\
&1 \text{ Compute } r_i \text{ for each } i \in F \text{ satisfying } f_i = \sum_{j \in C} \max \{ \min \{ r_i - c_{ij}, p_j - c_{ij} \}, 0 \}. \\
&/* \text{ Greedy Phase: } */ \\
&2 C' \leftarrow \emptyset, \quad F' \leftarrow \emptyset, \quad O' \leftarrow \emptyset. \\
&3 \text{ for } i = 0, 1, 2, \ldots \text{ do} \\
&4 \quad \text{ Let } W \text{ be the set of vertices } w \in F \text{ across all machines with } \tilde{r}_w = \tilde{r} = (1 + \varepsilon)^i. \\
&5 \quad \text{ Using Lemma } \text{ remove all vertices from } W \text{ within distance } 2(1 + \varepsilon)^2 \cdot \tilde{r} \text{ from } F'. \\
&6 \quad I \leftarrow \text{APPROXIMATEMIS}(G, W, 2(1 + \varepsilon)^3 \cdot \tilde{r}, \varepsilon). \\
&7 \quad F' \leftarrow F' \cup I. \\
&8 \text{ end} \\
&9 \text{ Using Lemma } \text{ add to } C' \text{ all clients } j \text{ having distance to } F' \text{ less than } p_j \text{ and add the rest to } O'. \\
&10 \text{ return } (C', F') \text{ as the solution.}
\end{align*}
\]

Our \( k \)-machine model implementation of the FacLoc with Penalties algorithm is summarized in Algorithm 10. The correctness proof is similar to that of Algorithm 3 but is complicated by the fact that we compute \((1 + \varepsilon)\)-approximate distances instead of exact distances. This analysis appears in the next section, and as a result we get the following theorem the proof of which is similar to Theorem 25.

\[ \text{Theorem 46. In } \tilde{O}(\text{poly}(1/\varepsilon) \cdot n/k) \text{ rounds, whp, Algorithm 10 finds a factor } 5 + O(\varepsilon) \text{ approximate solution } (C', F') \text{ to the FACLoc with Penalties problem.} \]

5.1.2 Analysis of the Algorithm

We state the standard primal and dual linear programming relaxations for facility location with penalties in Figure 2. For \( j \in C \) and \( i \in F \), define \( w_{ij} := \max \{ \min \{ r_i - c_{ij}, p_j - c_{ij} \}, 0 \} \)

and for \( j \in C \), let \( v'_j := \min_{i \in F} c_{ij} + w_{ij}. \) Note that \( v'_j = \min_{i \in F} \max \{ c_{ij}, \min \{ r_i, p_j \} \}. \) If \( i \)

is a facility realizing the minimum \( v'_j = c_{ij} + w_{ij} \), then we say that \( i \) is the bottleneck of \( j \).

To make the analysis easier, we consider a more costly solution \((\tilde{C}', F')\) where the set of clients \( C' \) is constructed using the following modified outlier determination phase: for each client \( j \), if \( c_{ij} \leq p_j \) then \( j \in \tilde{C}' \) and otherwise \( j \in \tilde{O}' \) where \( \tilde{O}' \) is the bottleneck of \( j \).

It is easy to see by an exchange argument that the \( \text{cost}_x(C', F') \leq \text{cost}_x(\tilde{C}', F') \), the outliers determined in the algorithm are at least as far from \( F' \) as ones in the modified outlier determination phase. Henceforth, we analyze the cost of the solution \((\tilde{C}', F')\) by comparing it to the cost of a feasible dual LP solution and in order to alleviate excessive notation, we will henceforth refer to the solution \((\tilde{C}', F')\) as \((C', F')\) and \( \tilde{O}' \) as \( O' \).
Because of the way we choose the outliers in the solution we consider for the analysis $(C',F')$ we have the following property (where $\iota$ is the bottleneck of $j$) –

$$v_j = \begin{cases} 
\max \{c_{ij}, r_i\} & \text{if } j \in C' \\
\rho_j & \text{if } j \in C \setminus C'
\end{cases}$$

Throughout this section, we condition on the event that the outcome of all the randomized algorithms is as expected (i.e. the “bad” events do not happen). Note that this happens with w.h.p. We first need the following facts along the lines of [39]. We skip the proofs as they are identical to the corresponding facts in the previous section.

### Lemma 47 (Modified From Lemma 8 Of [39]).
There exists a total ordering $<$ on the facilities in $F$ such that $u < v \implies \bar{\tau}_u \leq \bar{\tau}_v$, and $v$ is added to $F'$ if and only if there is no previous $u < v$ in $F'$ such that $c_{uv} \leq 2(1+\varepsilon)^2\bar{\tau}_v$.

### Claim 48 (Modified From Claim 9.2 Of [39]).
For any two distinct vertices $u, v \in F'$, we have that $c_{uv} > 2(1+\varepsilon)^2 \cdot \max\{\bar{\tau}_u, \bar{\tau}_v\}$.

We simultaneously prove feasibility of the dual solution we constructed, and show how it can be used to pay for the integral solution. We consider different cases regarding a fixed client $j \in C$ with bottleneck facility $\iota$. We first prove a few straightforward claims.

### Claim 49.
For any $i \in F$ and $j \in C$, $\bar{\tau}_i \leq (1+\varepsilon)^2(c_{ij} + w_{ij})$. Furthermore if for some $i \in F$ and $j \in C$, $w_{ij} > 0$, then $\bar{\tau}_i \geq (1+\varepsilon)^{-2}(c_{ij} + w_{ij})$.

**Proof.** We have $w_{ij} = \max\{0, \min\{r_i, p_j\} - c_{ij}\} \geq \min\{r_i, p_j\} - c_{ij}$ and therefore $\bar{\tau}_i \leq (1+\varepsilon)^2\bar{r}_i \leq (1+\varepsilon)^2(c_{ij} + w_{ij})$.

If for some $i \in F$ and $j \in C$, $w_{ij} > 0$, then $w_{ij} = r_i - c_{ij}$ which means that $\bar{\tau}_i \geq (1+\varepsilon)^{-2}\bar{r}_i = (1+\varepsilon)^{-2}(c_{ij} + w_{ij})$.

### Claim 50.
If $\iota \in F'$ is the bottleneck for $j \in C$, then $w_{i'j} = 0$ for all $i' \in F'$, where $i' \neq \iota$.

**Proof.** Suppose there exists a facility $i' \in F'$ with $w_{i'j} > 0$. That is, $\min\{r_{i'}, c_{i'j}, p_j - c_{i'j}\} > 0$, which further implies that $c_{i'j} < \min\{p_j, r_i\} \leq r_{i'} \leq (1+\varepsilon)^2\bar{r}_{i'}$.

If $c_{ij} \geq r_i$, then $v_{ij} = c_{ij} + \max\{0, \min\{r_i - c_{ij}, p_j - c_{ij}\}\} = c_{ij}$. However, since $w_{i'j} > 0$, $c_{ij} = v_{ij} \leq c_{ij} + w_{i'j} = \min\{p_j, r_{i'}\} \leq r_{i'} \leq (1+\varepsilon)^2\bar{r}_{i'}$.

Otherwise, $c_{ij} < r_i \leq (1+\varepsilon)^2\bar{r}_i$.

Therefore in either case, $c_{ij} \leq c_{ij} + w_{i'j} \leq 2(1+\varepsilon)^2 \max\{\bar{\tau}_{i'} , \bar{\tau}_j\}$, which is a contradiction to Claim 48.

### Claim 51.
If $\iota \notin F'$ is the bottleneck of $j \in C$ and $\max\{r_i, c_{ij}\} \leq p_j$, and $i' \in F'$ caused $\iota$ to close, then $c_{i'j} \leq 3(1+\varepsilon)^4 v_{i'}$.

**Proof.** Note that since we assume $\max\{r_i, c_{ij}\} \leq p_j$, we have $j \in C'$, $v_j = v_{i'}$, $c_{ij} \geq p_j$, and $r_i \leq p_j$. Since $i' \in F'$ caused $\iota$ to close, $\bar{\tau}_{i'} \leq \bar{\tau}_j$. Furthermore, $c_{i'j} \leq 2(1+\varepsilon)^2\bar{r}_{i'}$. Therefore, $c_{i'j} \leq c_{i'j} + c_{ij} \leq 2(1+\varepsilon)^3r_i + c_{ij} \leq 2(1+\varepsilon)^4r_i + c_{ij}$.

If $c_{ij} \geq r_i$, then $w_{ij} = 0$, and $c_{ij} = v_j$. This means $2(1+\varepsilon)^4r_i + c_{ij} \leq 3(1+\varepsilon)^4c_{ij} = 3(1+\varepsilon)^4v_j$ Otherwise, $c_{ij} < r_i$. Here, $v_j = \min\{r_i, p_j\}$ Then $2(1+\varepsilon)^4r_i + c_{ij} < 3(1+\varepsilon)^4r_i = 3(1+\varepsilon)^4 \min\{r_i, p_j\} \leq 3(1+\varepsilon)^4v_j$.

In either case, $c_{i'j} \leq 3(1+\varepsilon)^4 v_{i'}$.

### Claim 52.
If $\iota$ is the bottleneck of $j$, with $\max\{r_i, c_{ij}\} \leq p_j$, then $(1+\varepsilon)^2 v_{i'} \geq \bar{\tau}_i$.
Proof. Again, since we assume max \{r_i, c_{ij}\} \leq p_j, we have j \in C', v_j = v'_j, c_{ij} \geq p_j, and r_i \leq p_j.

Recall that v_j = v'_j = \max \{c_{ij}, \min \{r_i, p_j\}\} = \max \{c_{ij}, r_i\} \geq r_i \geq (1 + \varepsilon)^{-2} \tilde{r}_i and the claim follows.

We are now ready to prove the feasibility and approximation guarantee.

Lemma 53. The solution (v, w) is a feasible solution to the dual LP relaxation.

Proof. First note that constraints [13] and [16] are satisfied by construction for all i \in F and j \in C and so is constraint [14] for all j \in C'.

All that is left to show is that constraint [13] is satisfied for all j \in C'. Since j \in O', max \{r_i, c_{ij}\} > p_j.

We have, v_j := p_j < \max \{r_i, c_{ij}\} = \max \{c_{ij}, \min \{r_i, p_j\}\} = v'_j \leq c_{ij} + w_{ij} for any i \in F.

Lemma 54. For any j \in C', there is some i \in F' such that c_{ij} + w_{ij} \leq 3(1 + \varepsilon)^4 v_j.

Proof. In all the cases, we assume that max \{r_i, c_{ij}\} \leq p_j and therefore j \in C'. This also implies v_j = v'_j = \max \{c_{ij}, \min \{p_j, r_i\}\} = \max \{c_{ij}, r_i\}. Therefore, we can just disregard the penalties in the analysis.

Case 1. \(\epsilon \in F'\)

Connect j to \(\epsilon\). From Claim 50, we know that \(w_{ij} = 0\) for all other \(i' \in F'\).

1. If \(c_{ij} < r_i\), then \(v_j = c_{ij} + w_{ij}\). In this case, \(v_j\) pays for connecting \(j\) to \(\epsilon\) and also for \(j\)'s contribution to opening cost of \(\epsilon\) which is exactly \(w_{ij}\).

2. Otherwise \(c_{ij} \geq r_i\), then \(w_{ij} = 0\), which is \(j\)'s contribution towards \(\epsilon\). We have \(v_j = c_{ij}\) and therefore \(v_j\) pays for connecting \(j\) to \(\epsilon\).

Case 2. \(\epsilon \notin F'\) and \(w_{ij} = 0\) for all \(i \in F'\).

Let \(i'\) be the facility that caused \(\epsilon\) to close. Connect \(j\) to \(i'\). From Claim 51, we have \(c_{ij} < 3(1 + \varepsilon)^4 v_j\). Therefore, \(3(1 + \varepsilon)^4 v_j\) pays for the connection to \(i'\).

Case 3. \(\epsilon \notin F'\), there is some \(i' \in F'\) with \(w_{ij} > 0\), but \(i'\) did not cause \(\epsilon\) to close.

We connect \(j\) to \(i'\). By assumption \(w_{ij} = r_i - c_{ij} > 0\). Furthermore, let \(i\) be the facility that caused \(\epsilon\) to close. By Claim 51, we have \(c_{ij} \leq 3(1 + \varepsilon)^4 v_j\).

We have \(c_{ij} + w_{ij} = r_i\). Also, \(c_{ij} > 2(1 + \varepsilon)^2 \tilde{r}_{ij}\), by Claim 18.

Now, 2(c_{ij} + w_{ij}) = 2r_i \leq 2(1 + \varepsilon)^2 \tilde{r}_{ij} < c_{ij}' \leq c_{ij} + c_{ij}.

Subtracting \(c_{ij}\) from both sides, we get \(c_{ij} + 2w_{ij} \leq c_{ij} \leq 3(1 + \varepsilon)^4 v_j\). Therefore, \(3(1 + \varepsilon)^4 v_j\) pays for the connection cost of \(j\) to \(i'\) and also for (twice) \(j\)'s contribution towards opening of \(i'\).

Case 4. \(\epsilon \notin F'\) and \(i' \in F'\) with \(w_{ij} > 0\) caused \(\epsilon\) to close.

We connect \(j\) to \(i'\). From Claim 52, we have that \((1 + \varepsilon)^2 v_j \geq r_i\).

Since \(i'\) caused \(\epsilon\) to close, \(\tilde{r}_i \geq \tilde{r}_{ij} \geq (1 + \varepsilon)^{-2} r_i \geq (1 + \varepsilon)^{-2} c_{ij} + w_{ij}\).

Therefore, \(c_{ij} + w_{ij} \leq (1 + \varepsilon)^{2} \tilde{r}_i \leq (1 + \varepsilon)^4 r_i \leq (1 + \varepsilon)^4 v_j\). That is, \(3(1 + \varepsilon)^4 v_j\) pays for the connection cost of \(j\) to \(i'\), as well as its contribution towards opening of \(i'\).

Thus, \((v, w)\) is a feasible dual solution. We use the above analysis to conclude with the following theorem.
Theorem 55.

\[ \text{cost}(C', F') \leq 3(1 + \varepsilon)^4 \cdot \text{cost}(C^*, F^*). \]

Proof. We show \( \text{cost}(C', F') \leq 3(1 + \varepsilon)^4 \left( \sum_{j \in C} v_j \right) \), which is sufficient since \((v, w)\) is a feasible dual solution, and cost of any feasible dual solution is a lower bound on the cost of an integral optimal solution.

As we have argued previously, for any \( j \in C \setminus C' \), we have \( p_j = v_j \), and that for any \( j \in C' \), we have \( d(j, F') + s(j) \), where \( s(j) \geq 0 \) is the contribution of \( j \) towards opening a single facility in \( F' \). We have also argued that any \( j \in C' \) contributes \( s(j) \) for at most one open facility from \( F' \). It follows that,

\[
\text{cost}(C', F') = \sum_{i \in F'} f_i + \sum_{j \in C'} d(j, F') + \sum_{j \in C \setminus C'} p_j \\
= \left( \sum_{j \in C'} s(j) + d(j, F') \right) + \sum_{j \in C \setminus C'} p_j \\
\leq 3(1 + \varepsilon)^4 \sum_{j \in C'} v_j + \sum_{j \in C \setminus C'} v_j \\
\leq 3(1 + \varepsilon)^4 \left( \sum_{j \in C} v_j \right)
\]

5.2 The Congested Clique and MPC Algorithms

As argued in Section 3.2, the Congested Clique model is essentially the same as the \( k \)-machine model, where \( k = n \). Plugging this into Theorem 46, we get the following theorem.

Theorem 56. In \( O(\text{poly log } n) \) rounds of Congested Clique, whp, we can find a factor \( 3 + O(\varepsilon) \) approximate solution to the FacLoc with Penalties problem for any constant \( \varepsilon > 0 \).

In order to compute the radii of the facilities, the machines need to know the penalties of all the clients which can be done in \( O(1) \) rounds of MPC since each machine needs to receive \( n \) words corresponding to the penalties of each client. The rest of the MPC algorithm implementation is similar to the corresponding implementation for the Robust FacLoc problem (Section 5.2) so we don’t repeat them again. The only difference is that we are trying to implement Algorithm 10 instead. We summarize this result in the following theorem.

Theorem 57. In \( O(\text{poly log } n) \) rounds of MPC, whp, we can find a factor \( 3 + O(\varepsilon) \) approximate solution to the FacLoc with Penalties problem for any constant \( \varepsilon > 0 \).

6 Facility Location with Penalties: 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.
6.1 The Congested Clique Algorithm

In this section, we briefly sketch how to implement the Facility Location with Penalties algorithm from Section 2.2 in $O(\log \log \log n)$ rounds of the Congested Clique, in the explicit metric setting. Recall that in this setting, each vertex (i.e.) $v \in V$ knows $d(u, v)$ for all vertices $u \in V$.

At the beginning of the algorithm, each client $j \in C$ broadcasts its penalty $p_j$ — this takes $O(1)$ rounds. Once each facility $i \in F$ knows penalty $p_j$ of each client $j \in C$, it can locally compute $r_i$ satisfying $f_i = \sum_{j \in C} \max\{\min\{r_i - c_{ij}, p_j - c_{ij}\}, 0\}$. As argued in Section 2.2, it is without loss of generality to assume that for any facility $i \in F$ an $r_i$ satisfying this equation exists. This completes the radius computation phase.

The details of the greedy phase are similar to that from the Facility Location algorithms of [21, 23], (see also Section 4.1), where the computation of this phase is reduced to 3-ruling set computation. As argued in Section 4.1 this can be done in $O(\log \log \log n)$ rounds. In fact, for the Facility Location with Penalties problem, this is simpler since each vertex participates in at most one ruling set computation, as opposed to $O(\log n)$ different ruling sets as in the Robust Facility Location. It can be shown in a similar way that this results in an $O(1)$ approximation. We summarize our result in the following theorem.

\textbf{Theorem 58.} There is an $O(1)$-approximation algorithm in the Congested Clique model for FACLOC with Penalties, running in $O(\log \log \log n)$ rounds whp.

6.2 The MPC Algorithm

Since each vertex has explicit knowledge of $n$ distances, the overall memory is $O(n^2)$. Therefore, we can simulate the Congested Clique algorithm from the preceding section using Theorem 3.1 of [7] in the MPC model. We summarize our result in the following theorem.

\textbf{Theorem 59.} There is an $O(1)$-approximation algorithm for FACLOC with Penalties that can be implemented in the MPC model with $O(n)$ words per machine in $O(\log \log \log n)$ rounds whp.

7 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. [14], 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 $\Omega(n/k)$ 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. ISBN 0-89791-785-5. 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. ISBN 978-1-4503-2710-7. 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. ISBN 978-3-540-39658-1. 10.1007/978-3-540-39658-1_6.

4 S. Bandyapadhyay, T. Inamdar, S. Pai, and S. V. Pemmaraju. Near-Optimal Clustering in the k-machine model. *ArXiv e-prints*, October 2017.

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. ISBN 978-1-4503-2666-5. 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 Andrea 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. ISBN 978-3-95977-053-8. 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.

8 Andrew Berns, James Hegeman, and Siriram 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. 10.1007/978-3-642-31585-5_39.

9 Keren Censor-Hillel, Piotr Kaski, Janne H. Korhonen, Christoph Lenzen, Ani Paz, and Jukka Suomela. Algebraic Methods in the Congested Clique. In *Proceedings of the 2015
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. ISBN 0-89871-490-7. URL http://dl.acm.org/citation.cfm?id=365411.365555.

11 Edith Cohen. Size-Estimation Framework with Applications to Transitive Closure and Reachability. Journal of Computer and System Sciences, 55(3):441–453, 1997. ISSN 0022-0000. 10.1006/jcss.1997.1534. URL http://linkinghub.elsevier.com/retrieve/pii/S0022000097915348.

12 Jeffrey Dean and Sanjay Ghemawat. MapReduce: A Flexible Data Processing Tool. Commun. ACM, 53(1):72–77, January 2010. ISSN 0001-0782. 10.1145/1629175.1629198. URL http://doi.acm.org/10.1145/1629175.1629198.

13 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.

14 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. URL http://doi.acm.org/10.1145/2332432.2332443.

15 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. ISBN 978-1-4503-0813-7. 10.1145/2020408.2020515. URL http://doi.acm.org/10.1145/2020408.2020515.

16 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 Conference on Information and Knowledge Management, CIKM ’15, pages 273–282, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3794-6. 10.1145/2806416.2806508. URL http://doi.acm.org/10.1145/2806416.2806508.

17 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. ISBN 1-59593-452-9. 10.1145/1148109.1148152. URL http://doi.acm.org/10.1145/1148109.1148152.

18 Mohsen Ghaffari. Distributed MIS via All-to-All Communication. In Proceedings of the 5th ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017, pages 141–149, 2017. 10.1145/3087801.3087830. URL http://doi.acm.org/10.1145/3087801.3087830.

19 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. 10.1145/3212734.3212743. URL http://doi.acm.org/10.1145/3212734.3212743.
20 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. ISBN 978-3-642-25590-8. 10.1007/978-3-642-25591-5_39. URL http://dx.doi.org/10.1007/978-3-642-25591-5_39

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

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

23 James W. Hegeman, Sriram V. Pemmaraju, and Vivek Sardeshmukh. Near-Constant-Time Distributed Algorithms on a Congested Clique. *CoRR*, abs/1408.2071, 2014. URL http://arxiv.org/abs/1408.2071

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

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. ISSN 0004-5411. 10.1145/375827.375845. URL http://doi.acm.org/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. ISBN 978-1-61197-503-1. URL http://dl.acm.org/doi/10.1145/3174304.3175472

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. ISBN 978-0-89871-698-6. URL http://dl.acm.org/doi/10.1145/1873601.1873677

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. URL http://dl.acm.org/doi/10.1145/2722129.2722157

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. ISBN 978-1-4503-2065-8. 10.1145/2484239.2501983. URL http://doi.acm.org/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. ISBN 978-3-642-22011-1. URL http://dl.acm.org/doi/10.1145/2027223.2027230

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.
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. ISBN 978-1-4503-0032-2. 10.1145/1807167.1807184. URL http://doi.acm.org/10.1145/1807167.1807184

Ramgopal R. Mettu and C. Greg Plaxton. The Online Median Problem. SIAM J. Comput., 32(3):816–832, March 2003. ISSN 0097-5397. 10.1137/S0097539701383443. URL http://dx.doi.org/10.1137/S0097539701383443

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.

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. ISBN 978-1-4503-4210-0. 10.1145/2935764.2935785. URL http://doi.acm.org/10.1145/2935764.2935785

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. 10.1145/3210377.3210409. URL http://doi.acm.org/10.1145/3210377.3210409

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. ISBN 978-1-4503-0719-2. 10.1145/1993806.1993851. URL http://doi.acm.org/10.1145/1993806.1993851

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. ISSN 0360-0300. 10.1145/2522968.2522981. URL http://doi.acm.org/10.1145/2522968.2522981

Mikkel Thorup. Quick k-Median, k-Center, and Facility Location for Sparse Graphs. SIAM Journal on Computing, 34(2):405–432, 2005. ISSN 0097-5397. 10.1137/S0097539701388884.