Super-Fast Distributed Algorithms for Metric Facility Location

Andrew Berns, James Hegeman, and Sriram V. Pemmaraju*

Department of Computer Science
The University of Iowa
Iowa City, Iowa 52242-1419, USA
[andrew-berns,james-hegeman,sriram-pemmaraju]@uiowa.edu

1 Introduction

This paper explores the design of "super-fast" distributed algorithms in settings in which bandwidth constraints impose severe restrictions on the volume of information that can quickly reach an individual node. As a starting point of our exploration, we consider networks of diameter one (i.e., cliques) so as to focus on bandwidth constraints only and avoid penalties imposed by network distance between nodes. We assume the standard $\mathcal{CONGEST}$ model [17], which is a synchronous message-passing model in which each node in a size-n network can send a message of size $\mathcal{O}(\log n)$ along each incident communication link in each round. By "super-fast" algorithms we mean algorithms whose running time is strictly sub-logarithmic. Our working hypothesis is that in low-diameter settings, where congestion rather than network distance is the main bottleneck, we should be able to design algorithms that are much faster than algorithms for "local" problems in high diameter settings.

The focus of this paper is the *distributed facility location* problem which has been considered by several researchers [5,13,14,15] in low-diameter settings. We first describe the sequential version of the problem. The input to the facility location problem consists of a set of *facilities* $\mathcal{F} = \{x_1, x_2, \ldots, x_m\}$, a set of clients $\mathcal{C} = \{y_1, y_2, \ldots, y_n\}$, an *opening cost* f_i associated with each facility x_i, and a *connection cost* $D(x_i, y_j)$ between each facility x_i and client y_j. The goal is to find a subset $F \subseteq \mathcal{F}$ of facilities to *open* so as to minimize the facility opening cost plus connection costs, i.e.,

$$FacLoc(F) := \sum_{x_i \in F} f_i + \sum_{y_j \in \mathcal{C}} D(F, y_j),$$

where $D(F, y_j) := \min_{x_i \in F} D(x_i, y_j)$. Facility location is an old and well studied problem in operations research [1,3,8,9,19] that arises in contexts such as locating hospitals in a city or locating distribution centers in a region.

The *metric facility location* problem is an important special case of facility location in which the connection costs satisfy the following "triangle inequality:"

* This work is supported in part by National Science Foundation grant CCF 0915543

for any $x_i, x_{i'} \in \mathcal{F}$ and $y_j, y_{j'} \in \mathcal{C}$, $D(x_i, y_j)+D(y_j, x_{i'})+D(x_{i'}, y_{j'}) \geq D(x_i, y_{j'})$. The facility location problem, even in its metric version, is NP-complete and finding approximation algorithms for the problem has been a fertile area of research. A series of constant-factor approximation algorithms have been proposed for the metric facility location problem, with a steady improvement in the constant specifying the approximation factor. See [11] for a recent 1.488-approximation algorithm. This result is near-optimal because it is known [7] that the metric facility location problem has no polynomial time algorithm yielding an approximation guarantee better than 1.463 unless $NP \subseteq DTIME(n^{\mathcal{O}(\log \log n)})$. For non-metric facility location, a simple greedy algorithm yields an $\mathcal{O}(\log n)$-approximation and this is also optimal (to within a constant factor) because it easy to show that the problem is at least as hard as set cover.

More recently, the facility location problem has also been used as an abstraction for the problem of locating resources in wireless networks [4,16]. Motivated by this application, several researchers have considered the facility location problem in a distributed setting. In [13,14,15] the underlying communication network is the complete bipartite graph with $\mathcal{F}$ forming one part and $\mathcal{C}$ forming the other part. At the beginning of the algorithm each node, whether it is a facility or a client, knows connection costs between itself and all nodes in the other part. In addition, the facilities know their opening costs. In [5] the underlying communication network is a clique. Each node in the clique may choose to open as a facility and each node that is not open will connect to an open facility. Note that all of this work assumes the $\mathcal{CONGEST}$ model of distributed computation. The facility location problem considered in [16] assumes that the underlying communication network is a *unit disk graph* (UDG). While such a network can have high diameter relative to the number of nodes in the network, this paper [16] reduces the UDG facility location problem to a low-diameter facility location-type problem and uses this in the final solution. In summary, there has been a fair bit of research on distributed facility location in low-diameter settings.

None of the prior papers, however, achieve near-optimal approximation (i.e., constant-factor in the case of metric facility location and $\mathcal{O}(\log n)$ in the case of non-metric facility location) in *sub-logarithmic* rounds. While [5] does present a *constant-round*, constant-factor approximation to metric facility location on a clique, it is only for the special case of *uniform* metric facility location, i.e., when all facility opening costs are identical. The question that drives this paper, then, is the following: can we achieve a distributed constant-factor approximation algorithm for the metric facility location problem in the clique setting in strictly sub-logarithmic time? One can ask similar questions in the bipartite setting and for non-metric facility location as well, but as a first step we focus on the metric version of the facility location problem on a clique.

Distributed facility location is challenging even in a low-diameter setting because the input consists of $\Theta(m \cdot n + m)$ pieces of information, distributed across the network, which cannot quickly be delivered to a single node (or even a small number of nodes) due to the bandwidth constraints of the $\mathcal{CONGEST}$ model. Therefore, any super-fast distributed algorithm for the problem must be truly

distributed and needs to take advantage of the available bandwidth and of structural properties of approximate solutions to the metric facility location problem. Also worth noting is that even though tight lower bounds on the running time of distributed approximation algorithms have been established [10], none of these bounds extend to low-diameter settings.

Main result. The main result of this paper is an $\mathcal{O}(\log\log n \cdot \log^* n)$-round, $\mathcal{O}(1)$-approximation algorithm for metric facility location on a clique. If the metric satisfies additional properties (e.g., it has constant doubling dimension) then we obtain an $\mathcal{O}(\log^* n)$-round $\mathcal{O}(1)$-approximation to the problem. Our results are achieved via a combination of techniques that include (i) a new lower bound for the optimal cost of metric facility location and (ii) a sparsification technique combining randomization with a deterministic subroutine that repeatedly leverages the available bandwidth to process sparse subgraphs. These technical contributions are described in the following section in greater detail.

1.1 Overview of Technical Contributions

We start by describing the distributed facility location problem on a clique, as in [6]. Let (P, D) be a discrete metric space with point set $P = \{p_1, p_2, \ldots, p_n\}$. Let f_i be the opening cost of p_i. Now view the metric space (P, D) as a completely connected size-n network $C = (P, E)$ with each point p_i represented by a node, which we will also call p_i. Each node p_i knows f_i and the connection costs (distances) $D(p_i, p_j)$ for all $p_j \in P$. The problem is to design a distributed algorithm that runs on C in the $\mathcal{CONGEST}$ model and produces a subset $X \subseteq P$ such that each node $p_i \in X$ opens and provides services as a facility and each node $p_i \notin X$ connects to the nearest open node. The goal is to guarantee that $FacLoc(X) \leq \alpha \cdot OPT$, where OPT is the cost of an optimal solution to the given instance of facility location and α is some constant. We call this the CliqueFacLoc problem. Of course, we also want our algorithm to be "super-fast," i.e., terminate in $o(\log n)$ rounds.

Our paper makes three main technical contributions. We summarize these as follows.

- **A new lower bound for metric facility location.** For $p \in P$, let $B(p, r)$ denote the set of points $q \in P$ satisfying $D(p, q) \leq r$. For each p_i, let r_i be the non-negative real number satisfying

$$\sum_{q \in B(p_i, r_i)} (r_i - D(p_i, q)) = f_i$$

  As observed by Mettu and Plaxton [12], r_i exists and is unique.
  Bădoiu et al. proved in [2] that $\sum_{i=1}^{n} r_i$ is a constant-factor approximation for OPT in the case of *uniform* facility opening costs. This fact plays a critical role in the design of the constant-round, constant-factor approximation algorithm of Gehweiler et al. [6] for the special case of CliqueFacLoc in

which all facility opening costs are identical. Unfortunately, the sum $\sum_{i=1}^{n} r_i$ can be arbitrarily large compared to OPT when the f_i's are allowed to vary. Consider an example consisting of only two nodes, one of whose opening costs is astronomical in comparison to the other and to the distance between them. The optimal solution to this problem consists of opening only one facility - the one with small opening cost. However, note that the r-value for the facility with an astronomical opening cost is also astronomical.
Therefore, we apply the idempotent transformation

$$r_i \rightarrow \bar{r}_i = \min_{1 \leq j \leq n} \{D(p_i, p_j) + r_j\},$$

and use $\bar{r}_i$ instead of r_i to derive a lower bound. Note that for all i, $\bar{r}_i \leq r_i$. We will show later in the paper that $\sum_{i=1}^{n} \bar{r}_i$ does approximate the optimal cost OPT to within a constant factor in the general case of non-uniform facility opening costs. This turns out to be a key ingredient of our approach.

– **Reduction to an $\mathcal{O}(1)$-ruling set.** Our next contribution is an $\mathcal{O}(1)$-round reduction of the distributed facility location problem on a clique to the problem of computing an $\mathcal{O}(1)$-ruling set. To be more specific let $C' = (P, E')$ be a spanning subgraph of C. A subset $Y \subseteq P$ is said to be *independent* if no two nodes in Y are neighbors in C'. An independent set Y is a *maximal independent set* (MIS) if no superset $Y' \supset Y$ is independent in C'. An independent set Y is β-ruling if every node in P is at most β hops along edges in C' from some node in Y. Clearly, an MIS is a 1-ruling set.
We describe an algorithm that solves distributed facility location on a clique by first computing a collection of subgraphs $C_1, C_2, C_3, \ldots$ (such that $\bigcup C_i = C$) in $\mathcal{O}(1)$ rounds and then a further collection of spanning subgraphs $C'_1, C'_2, C'_3, \ldots$ (of $C_1, C_2, C_3, \ldots$), also in $\mathcal{O}(1)$ rounds. Then we show that a solution to the facility location problem (i.e., a set of nodes to open) can be obtained by computing a β-ruling set for each of the spanning subgraphs C'_j, $j \geq 1$, and combining the solutions in a certain way. We show that combining the β-ruling sets can also be done in $\mathcal{O}(1)$ rounds. The parameter β affects the approximation factor of the computed solution and enforcing $\beta = \mathcal{O}(1)$ ensures that the solution to facility location is an $\mathcal{O}(1)$-approximation.

– **An $\mathcal{O}(1)$-ruling set via a combination of randomized and deterministic sparsification.** We present an $\mathcal{O}(\log\log n \cdot \log^* n)$-round algorithm to compute a 2-ruling set of a given spanning subgraph C' of C. We see this as our main contribution and believe that our approach will be useful in general for developing "super-fast" algorithms in low diameter settings. We start by describing a deterministic "subroutine" that takes a subset $Z \subseteq P$ as input and computes an MIS of $C'[Z]$ (i.e., the subgraph of C' induced by Z) in c rounds if $C'[Z]$ has at most $c \cdot n$ edges. This is achieved via a simple load balancing scheme that communicates the entire subgraph $C'[Z]$ to all nodes in c rounds. Then we show how to use randomization repeatedly to peel off subgraphs with linearly many edges for processing by the aforementioned subroutine. We show that, in this manner, the entire graph C' can be processed using $\mathcal{O}(\log\log n \cdot \log^* n)$ calls to the deterministic subroutine.

2 Reduction to the $\mathcal{O}(1)$-Ruling Set Problem

2.1 A New Lower Bound for Non-Uniform Metric Facility Location

In this subsection we show that $\sum_{i=1}^{n} \bar{r}_i$ is a constant-factor lower bound to the optimal cost OPT. To facilitate this, we recall a definition from Mettu and Plaxton [12]. The *charge* of a node p_i with respect to a collection of (open) facilities X (also known as a *configuration*) is defined by

$$charge(p_i, X) = D(p_i, X) + \sum_{p_j \in X} \max\{0, r_j - D(p_j, p_i)\}$$

where $D(p_i, X) = \min_{p_j \in X} D(p_i, p_j)$. Mettu and Plaxton showed that the cost of a configuration X, $FacLoc(X)$, is precisely equal to the sum of the charges with respect to X, i.e. $\sum_{i=1}^{n} charge(p_i, X)$ [12].

The Mettu-Plaxton configuration F_{MP}, derived from the (sequential) Mettu-Plaxton algorithm [12], was shown to have a cost at most three times OPT. So for any configuration X, $FacLoc(X) \geq \frac{1}{3} FacLoc(F_{MP}) = \frac{1}{3} \sum_{i=1}^{n} charge(p_i, F_{MP})$.

We now present the following lemma, which relates $FacLoc(X)$ (for any X) to $\sum_{i=1}^{n} \bar{r}_i$. Due to space constraints, the proof of all lemmas and theorems are included in Appendix A.

Lemma 1. *$FacLoc(X) \geq (\sum_{i=1}^{n} \bar{r}_i)/6$ for any configuration X.*

2.2 Algorithm

We present our facility location algorithm in Algorithm 1. Our deterministic distributed algorithm consists of three stages. We use the notations $G[H]$ and $E[G]$ to refer to the subgraph induced by H and the edge set of G, respectively.

Stage 1 *(Steps 1-2).* Each node knows its own opening cost and the distance to other nodes, so in Step 1 node p_i computes r_i and broadcasts that value to all others. (We acknowledge the implicit assumption here that the number of bits needed to represent a given distance or opening cost in the network does not exceed $\mathcal{O}(\log n)$.) Once this broadcast is complete, each node knows all of the r_i values. This enables every node to compute a partition of the network into groups whose r_i values vary by at most a factor of 3 (Step 2). More specifically, define the special value $r_0 := \min_{1 \leq j \leq n}\{r_j\}$. Define the class H_k to be the set of nodes p_i such that $3^k \cdot r_0 \leq r_i < 3^{k+1} \cdot r_0$. Every node computes the class into which each node in the network, including itself, falls.

Stage 2 *(Steps 3-5).* Now that the nodes are divided into classes having comparable r_i's, and every node knows who is in each class, we focus our attention on class H_k. Suppose $p_i, p_j \in H_k$. Then $3^k \cdot r_0 \leq r_i, r_j < 3^{k+1} \cdot r_0$, and we define p_i and p_j to be *adjacent* in class H_k if $D(p_i, p_j) \leq r_i + r_j$. Each node in class H_k can determine its neighbors in H_k. Next, compute a sparse set $T_k \subseteq H_k$

with procedure RulingSet(). We describe a super-fast implementation of RulingSet() in Section 3. After a sparse set has been constructed for each class H_k, each node broadcasts whether or not it has been included in the sparse set of its own class, and so at the end of Stage 2, every node knows the members of the sparse set in each class.

Stage 3 *(Steps 6-7).* Finally, a node p_i in class H_k declares itself to be open if (i) $p_i \in T_k$, and (ii) there is no node $p_j \in B(p_i, 2r_i)$ of class $H_{k'}$ for which $k' < k$. Lastly, open facilities declare themselves as such in a broadcast, and every node connects to the nearest open facility.

Algorithm 1 FacilityLocation

Input: A discrete metric space of nodes (P, D), with opening costs; a sparsity parameter s

Assumption: Each node knows its own opening cost and the distances from itself to other nodes

Output: A subset of nodes (a *configuration*) to be declared open, whose cost is a constant-factor approximation to the optimal cost

1. Each node p_i computes and broadcasts its value r_i; $r_0 := \min_i r_i$
2. Each node computes a partition of the network into classes H_k with $3^k \cdot r_0 \leq r_j < 3^{k+1} \cdot r_0$ for $p_j \in H_k$
3. Each node determines its neighbors within its own class; for $p_i, p_j \in H_k$, $(p_i, p_j) \in E[G[H_k]]$ iff $D(p_i, p_j) \leq r_i + r_j$
4. For each H_k, facilities in class H_k use procedure RulingSet$(G[H_k], s)$ to determine a sparse set $T_k \subseteq H_k$
5. Each node broadcasts its membership status with respect to the sparse set T_k of its own class
6. A node $p_i \in H_k$ declares itself to be open if both of the following conditions hold:
   (i) p_i is a member of the sparse set $T_k \subseteq H_k$
   (ii) There is no node p_j belonging to a class $H_{k'}$, with $k' < k$, such that $D(p_i, p_j) \leq 2r_i$
7. Each node broadcasts its status (open or not), and every node connects to the nearest open node

2.3 Analysis

We show that our algorithm produces an $\mathcal{O}(s)$-approximation to CliqueFacLoc in $\mathcal{O}(\mathcal{T}(n))$ communication rounds, where $\mathcal{T}(n)$ is the running time (in rounds) of procedure Ruling-Set().

Algorithm 2 RulingSet

Input: An undirected graph $G = (V, E)$; a sparsity parameter s
Assumptions: Each node has knowledge of its neighbors in G;
each node can send a message to any other node (not just along edges of G)
Output: An independent s-ruling set $T \subseteq G$

Communication rounds. Stage 1 requires exactly one round of communication, to broadcast r_i values. Stage 2 requires $\mathcal{O}(\mathcal{T}(n))$ rounds to compute the s-sparse subsets $\{T_k\}_k$, and an additional round to broadcast membership status. Stage 3 requires one round, in order to inform others of a nodes decision to open or not. Thus, the running time of our algorithm in communication rounds is $\mathcal{O}(\mathcal{T}(n))$.

Cost approximation. Let F be the set of nodes opened by our algorithm. We analyze $FacLoc(F)$ by bounding $charge(p_i, F)$ for each p_i. Recall that $FacLoc(F) = \sum_{i=1}^{n} charge(p_i, F)$. Since $charge(p_i, F)$ is the sum of two terms, $D(p_i, F)$ and $\sum_{p_j \in F} \max\{0, r_j - D(p_j, p_i)\}$, we bound each separately by a $\mathcal{O}(s)$-multiple of $\bar{r}_i$.

The sparse subset $T_k \subseteq H_k$ has the property that for any node $p_i \in H_k$, $D(p_i, T_k) \leq 2 \cdot 3^{k+1} r_0 \cdot s$, where s is the sparsity parameter passed to procedure RulingSet(). Also, for no two members of T_k is the distance between them less than $2 \cdot 3^k r_0$. Note that here we are using distances from the metric D of (P, D).

Now, in our cost analysis, we consider a node $p_i \in H_k$. To bound $D(p_i, F)$, observe that either $p_i \in T_k$, or else there exists a node $p_j \in T_k$ such that $D(p_i, p_j) \leq 2 \cdot 3^{k+1} r_0 \cdot s \leq 6 r_i \cdot s$. Also, if a node $p_j \in T_k$ does not open, then there exists another node $p_{j'}$ in a class $H_{k'}$, with $k' < k$, such that $D(p_j, p_{j'}) \leq 2 r_j$.

We are now ready to bound the components of $charge(p_i, F)$.

Lemma 2. $D(p_i, F) \leq (81s + 81) \cdot \bar{r}_i$

Lemma 3. $\sum_{p_j \in F} \max\{0, r_j - D(p_j, p_i)\} \leq 9\bar{r}_i$

Combining the two previous lemmas gives

$$FacLoc(F) = \sum_{i=1}^{n} charge(p_i, F) = \sum_{i=1}^{n} \left[D(p_i, F) + \sum_{p_j \in F} \max\{0, r_j - D(p_j, p_i)\} \right]$$
$$\leq \sum_{i=1}^{n} \left[(81s + 81) \cdot \bar{r}_i + 9\bar{r}_i \right] \leq (81s + 90) \cdot \sum_{i=1}^{n} \bar{r}_i$$

Theorem 1. *Algorithm 1 (*FacilityLocation*) computes an $\mathcal{O}(s)$-factor approximation to* CliqueFacLoc *in $\mathcal{O}(\mathcal{T}(n))$ rounds, where $\mathcal{T}(n)$ is the running time of the* RulingSet*() procedure called with argument s.*

3 Computing a 2-Ruling Set

The facility location algorithm in Section 2 depends on being able to efficiently compute an independent β-ruling set, for small β, of an arbitrary spanning subgraph C' of the clique C. This section describes how to compute an independent 2-ruling set of C' in $\mathcal{O}(\log\log n \cdot \log^* n)$ rounds.

3.1 A Useful Subroutine

We first present an extremely useful deterministic subroutine for efficiently computing an MIS of a sparse, induced subgraph of C'. For a subset $M \subseteq P$, we use $C'[M]$ to denote the subgraph of C' induced by M and $E[M]$ and $e[M]$ to denote the set and number (respectively) of edges in $C'[M]$. The subroutine we present below computes an MIS of $C'[M]$ in $e[M]/n$ rounds. Later, we use this repeatedly in situations where $e[M]$ is linear in n.

We assume that nodes in P have unique identifiers and can therefore be totally ordered according to these. Let $\rho_i \in \{0, 1, \ldots, n-1\}$ denote the rank of node p_i in this total ordering. Imagine (temporarily) that edges are oriented from lower rank nodes to higher rank nodes and let $\mathcal{E}(p_i)$ denote the set of outgoing edges incident on p_i. Let d_i denote $|\mathcal{E}(p_i)|$, the outdegree of p_i, and let $D_i = \sum_{j:\rho_j<\rho_i} d_j$ denote the outdegree sum of lower ranked nodes.

The subroutine works by sharing the entire topology of $C'[M]$ with all nodes in the network. To do this efficiently, we "map" each edge $e \in E[M]$ to a node in P. Information about e will be sent to the node to which e is mapped and then that node is responsible for broadcasting information about all edges that have been mapped onto it. If the mapping is such that the "load" at each node is balanced, then we have an efficient algorithm. Our subroutine is given below in Algorithm 3.

Algorithm 3 Deterministic MIS for Sparse Graphs

Input: A subset of nodes $M \subseteq P$
Output: An MIS L of $C'[M]$
Algorithm executed by node p_i

1. Broadcast ID.
2. Calculate d_i and broadcast it.
3. Assign a distinct label $\ell(e)$ from $\{D_i, D_i+1, \ldots, D_i+d_i-1\}$ to each incident outgoing edge e.
4. Send each outgoing edge e to a node p_j with rank $\rho_j = \ell(e)(\mod n)$.
5. Broadcast all edges received in previous step, one per round.
6. Compute $C'[M]$ from received edges and use a deterministic algorithm to locally compute MIS L.

Theorem 2. *Algorithm 3 computes an MIS L of $C'[M]$ in $\frac{e[M]}{n} + \mathcal{O}(1)$ rounds.*

3.2 Algorithm

We are now ready to present a "super-fast" algorithm for computing a 2-ruling set of C'. We will show that this algorithm terminates in $\mathcal{O}(\log\log n \cdot \log^* n)$ rounds. The algorithm proceeds in *Stages* and in Stage i, $i = 1, 2, \ldots$ we process nodes whose degrees (in graph C') lie in the range $[n^{1/2^i}, n^{1/2^{i-1}})$. At the end of Stage i all nodes in the graph have degree less than $n^{1/2^i}$, implying the algorithm consists of $\mathcal{O}(\log\log n)$ Stages.

Each Stage consists of a number of *Phases*. Suppose that we are in a Stage where we are processing the set $S(d)$ of nodes whose degrees are in the range $[d, d^2)$. In each Phase of this Stage, the size of $S(d)$ decreases. To understand the rate at which this happens, consider the function $t(k)$ defined recursively as $t(0) = 1$, and $t(k+1) = e^{\sqrt{t(k)}}$, for all $k > 0$. As we will show later, this is a rapidly-growing function that reaches n in $\mathcal{O}(\log^* n)$ steps. The size of $S(d)$ at the beginning of Phase k is at most $n/t(k)$ and as the Phase proceeds, $S(d)$ shrinks and Phase k ends when $|S(d)| \leq \frac{n}{t(k+1)}$ (**while**-loop starting in Line 6). Because of the rate at which $t(k)$ grows, each Stage consists of $\mathcal{O}(\log^* n)$ Phases.

Each Phase consists of a number of *Iterations*. Consider a Stage in which nodes whose degrees are in $[d, d^2)$ are being processed and then consider Phase k in this Stage. Finally, consider an arbitrary Iteration in this Phase. In this Iteration, nodes in $S(d)$ join a set M independently with probability $q = \sqrt{t(k)}/d$ (Line 8). Nodes not in $S(d)$ join M with probability $1/\sqrt{d}$ (Line 9) (note that this probability is independent of k and is therefore fixed for the entire Stage). The function $t(k)$ was designed to yield probabilities q such that the expected number of edges in $C'[M]$ is bounded above by $2n$. Once the set M is picked, we call the subroutine in Algorithm 3 to process $C'[M]$ in $\mathcal{O}(1)$ rounds and then delete M and its neighborhood $N(M)$ (Lines 10-12). This ends an Iteration. We will show that in expectation, only a constant number of Iterations are needed to complete a Phase. Since the size of $S(d)$ has diminished in a phase, we can afford to raise the probability q (Line 15) and still ensure that the expected number of edges in $C'[M]$ is bounded above by $2n$. Within a Stage the probability q rises until it gets to $1/\sqrt{d}$. Recall that this is the probability with which the nodes not in $S(d)$ include themselves in M. We will show below that by the time q reaches $1/\sqrt{d}$, the size of $S(d)$ will have decreased so that the subgraph induced by $S(d)$ contains only $\mathcal{O}(n)$ edges. We can separately process this subgraph in $\mathcal{O}(1)$ time (Lines 18-20) to finish the Stage. The full algorithm can be found in Algorithm 4.

3.3 Analysis

We start by stating the following lemma concerning the behavior of $t(k)$.

Lemma 4. *For any $d \geq 0$, the smallest k for which $t(k) \geq d$ is $\mathcal{O}(\log^* d)$.*

Algorithm 4 Super-Fast 2-Ruling Set

Input: A spanning subgraph C' of the clique C.
Output: A 2-ruling set $T \subseteq P$ of C'

1. $i = 1$; $d := \sqrt{n}$ $(= n^{1/2^i})$; $T := \emptyset$
2. **while** $d > 10$ **do**
   Start of Stage i***:***
3. $k := 0$; $q := \frac{1}{d}$ $(= \sqrt{t(k)}/d)$;
4. $S(d) := \{p \in P : deg(p) \geq d\}$; $lastPhase := false$;
5. **while** $(true)$ **do**
   Start of Phase k***:***
6. **while** $(|S(d)| > \frac{n}{t(k+1)}$ **and** $\neg lastPhase)$
   or $(|S(d)| > \frac{n}{e^{\sqrt{d}}}$ **and** $lastPhase)$ **do**
   Start of Iteration
7. $M := \emptyset$
8. Add each $p \in S(d)$ to M with probability q
9. Add each $p \in P \setminus S(d)$ to M with probability $\frac{1}{\sqrt{d}}$
10. Compute an MIS L on M using Algorithm 3
11. $T := T \cup L$
12. Remove $(M \cup N(M))$ from C'
13. $S(d) := \{p \in P : deg(p) \geq d\}$
   End of Iteration
14. **if** $lastPhase$ **then break**;
15. $q := \frac{\sqrt{t(k+1)}}{d}$; $k := k + 1$;
16. **if** $q > \frac{1}{\sqrt{d}}$ **then**
17. $q := \frac{1}{\sqrt{d}}$; $lastPhase := true$;
   End of Phase
18. $M := S(d)$; Compute an MIS L on M using Algorithm 3
19. $T := T \cup L$
20. Remove $(M \cup N(M))$ from C'
21. $d := n^{1/2^{i+1}}$; $i := i + 1$
   End of Stage
22. $M := C'$; Compute an MIS L on M using Algorithm 3
23. $T := T \cup L$
24. Output T

Lemma 5. *Consider an Iteration in Phase k, Stage i. Let $d = n^{1/2^i}$. The maximum degree of a node during this Iteration is d^2. Furthermore, the size of $S(d)$ is at most $n/t(k)$.*

Lemma 6. *Algorithm 4 computes a 2-ruling set of C'.*

Lemma 7. *In any Iteration, the expected number of edges in the subgraph induced by M is bounded above by $2n$.*

Lemma 8. *Fix a Stage i, and suppose that $t(k+1) \leq d$. Then the expected number of Iterations (Lines 7-13) in Phase k before $|\mathcal{S}(d)| \leq \frac{n}{t(k+1)}$ is $\mathcal{O}(1)$.*

Lemma 9. *Fix a Stage i, and suppose $t(k) \leq d < t(k+1)$. At the end of Phase $k+1$, there are at most $\mathcal{O}(n)$ edges in $C'[S(d)]$.*

Theorem 3. *Algorithm 4 computes a 2-ruling set on C' in $\mathcal{O}(\log\log n \cdot \log^* n)$ rounds.*

4 Wrapping Things Up

Using Algorithm 4 as a specific instance of the procedure RulingSet() for $s = 2$ and combining Theorems 1 and 3 leads us to the main result of the paper.

Theorem 4. *The* CliqueFacLoc *problem can be solved in $\mathcal{O}(\log\log n \cdot \log^* n)$ rounds.*

We also note that under special circumstances an $\mathcal{O}(1)$-ruling set of a spanning subgraph of a clique can be computed even more speedily. For example, if the subgraph of C induced by the nodes in class H_k is growth-bounded for each k, then we can use the Schneider-Wattenhofer [18] result to compute an MIS for $C[H_k]$ in $\mathcal{O}(\log^* n)$ rounds (in the $\mathcal{CONGEST}$ model). Recall that edges in H_k connect pairs of nodes whose distances in the metric space (P, D) are roughly the same (i.e. within a factor of 3). It is easy to see that if the metric space (P, D) is Euclidean with constant dimension or even has constant doubling dimension, H_k would be growth-bounded for each k. This discussion is summarized in the following theorem.

Theorem 5. *The* CliqueFacLoc *problem can be solved in $\mathcal{O}(\log^* n)$ rounds on a metric space of constant doubling dimension.*

References

1. Balinski, M.L.: On finding integer solutions to linear programs. In: Proceedings of IBM Scientific Computing Symposium on Combinatorial Problems. pp. 225–248 (1966)
2. Bădoiu, M., Czumaj, A., Indyk, P., Sohler, C.: Facility location in sublinear time. In: ICALP. pp. 866–877 (2005)
3. Cornuejols, G., Nemhouser, G., Wolsey, L.: Discrete Location Theory. Wiley (1990)

4. Frank, C.: Algorithms for Sensor and Ad Hoc Networks. Springer (2007)
5. Gehweiler, J., Lammersen, C., Sohler, C.: A distributed o(1)-approximation algorithm for the uniform facility location problem. In: SPAA '06: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures. pp. 237–243 (2006)
6. Gehweiler, J., Lammersen, C., Sohler, C.: A distributed o(1)-approximation algorithm for the uniform facility location problem. In: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures. pp. 237–243. SPAA '06, ACM, New York, NY, USA (2006), http://doi.acm.org/10.1145/1148109.1148152
7. Guha, S., Khuller, S.: Greedy strikes back: improved facility location algorithms. In: SODA '98: Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms. pp. 649–657 (1998)
8. Kaufman, L., Eede, M.V., Hansen, P.: A plant and warehouse location problem. Operational Research Quarterly 28(3), 547–554 (1977)
9. Kuehn, A.A., Hamburger, M.J.: A heuristic program for locating warehouses. Management Science 9(4), 643–666 (1963)
10. Kuhn, F., Moscibroda, T., Wattenhofer, R.: Local computation: Lower and upper bounds. CoRR abs/1011.5470 (2010)
11. Li, S.: 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. pp. 77–88. ICALP'11, Springer-Verlag, Berlin, Heidelberg (2011), http://dl.acm.org/citation.cfm?id=2027223.2027230
12. Mettu, R.R., Plaxton, C.G.: The online median problem. SIAM J. Comput. 32(3), 816–832 (2003)
13. Moscibroda, T., Wattenhofer, R.: Facility location: distributed approximation. In: PODC '05: Proceedings of the Twenty-Fourth Annual ACM Symposium on Principles of Distributed Computing. pp. 108–117 (2005)
14. Pandit, S., Pemmaraju, S.V.: Rapid randomized pruning for fast greedy distributed algorithms. In: ACM Symp. on Principles of Distributed Computing (PODC). pp. 325–334 (2010)
15. Pandit, S., Pemmaraju, S.: Return of the primal-dual: distributed metric facilitylocation. In: Proceedings of the 28th ACM symposium on Principles of distributed computing. pp. 180–189. PODC '09, ACM, New York, NY, USA (2009), http://doi.acm.org/10.1145/1582716.1582747
16. Pandit, S., Pemmaraju, S.V.: Finding facilities fast. In: ICDCN. pp. 11–24 (2009)
17. Peleg, D.: Distributed computing: a locality-sensitive approach. Society for Industrial and Applied Mathematics (2000)
18. Schneider, J., Wattenhofer, R.: A log-star distributed maximal independent set algorithm for growth-bounded graphs. In: PODC. pp. 35–44 (2008)
19. Stollsteimer, J.F.: A working model for plant numbers and locations. Management Science 45(3), 631–645 (1963)

A Proofs

Lemma 1. *$FacLoc(X) \geq (\sum_{i=1}^{n} \bar{r}_i)/6$ for any configuration X.*

Proof. Notice F_{MP} has the property that no two facilities $p_i, p_j \in F_{MP}$ can be close enough that $D(p_i, p_j) \leq r_i + r_j$ [12]. Therefore, if $p_{\delta(i)}$ denotes a closest open facility (i.e. an open facility satisfying $D(p_i, p_{\delta(i)}) = D(p_i, F_{MP})$), then

$$\begin{aligned} FacLoc(F_{MP}) &= \sum_{i=1}^{n} charge(p_i, F_{MP}) \\ &= \sum_{p_j \in F_{MP}} charge(p_j, F_{MP}) + \sum_{p_i \notin F_{MP}} charge(p_i, F_{MP}) \\ &\geq \sum_{p_j \in F_{MP}} r_j + \sum_{p_i \notin F_{MP}} \left[D(p_i, p_{\delta(i)}) + \max\{0, r_{\delta(i)} - D(p_{\delta(i)}, p_i)\} \right] \\ &= \sum_{p_j \in F_{MP}} r_j + \sum_{p_i \notin F_{MP}} \max\{r_{\delta(i)}, D(p_i, p_{\delta(i)})\} \end{aligned}$$

Note that the inequality in the above calculation follows from observing that $charge(p_j, F_{MP}) \geq r_j$ for $p_j \in F_{MP}$, and from throwing away some terms of the sum in the definition of $charge(p_i, F_{MP})$ for $p_i \notin F_{MP}$.

Now, recall the definition $\bar{r}_i = \min_{1 \leq x \leq n}\{D(p_i, p_x) + r_x\}$. Therefore, $\bar{r}_i \leq r_i$, and $\bar{r}_i \leq D(p_i, p_{\delta(i)}) + r_{\delta(i)} \leq 2 \cdot \max\{r_{\delta(i)}, D(p_i, p_{\delta(i)})\}$. It follows that

$$\begin{aligned} FacLoc(F_{MP}) &\geq \sum_{p_j \in F_{MP}} \bar{r}_j + \sum_{p_i \notin F_{MP}} \frac{\bar{r}_i}{2} \\ &\geq \sum_{p_j \in F_{MP}} \frac{\bar{r}_j}{2} + \sum_{p_i \notin F_{MP}} \frac{\bar{r}_i}{2} \\ &= \frac{1}{2} \cdot \sum_{i=1}^{n} \bar{r}_i \end{aligned}$$

Therefore $FacLoc(X) \geq FacLoc(F_{MP})/3 \geq (\sum_{i=1}^{n} \bar{r}_i)/6$, for any configuration X.

Lemma 2. *$D(p_i, F) \leq (81s + 81) \cdot \bar{r}_i$*

Proof. We use induction on k. For $k = 0$, first note that if $p_i \in H_0$, then $r_i \leq 3\bar{r}_i$, as $r_i < 3r_0$. Secondly, observe that every member of T_0 must open, and so $D(p_i, F) \leq 2 \cdot 3r_0 \cdot s \leq 6r_i \cdot s \leq 18\bar{r}_i \cdot s$.

For the induction step, first suppose that $r_i > 3\bar{r}_i$. Let $p_{i'}$ be a minimizer for $D(p_i, p_x) + r_x$, so that $\bar{r}_i = D(p_i, p_{i'}) + r_{i'}$. Then $p_{i'}$ must be in a class of index

strictly less than k. Therefore, we may apply the induction assumption to $p_{i'}$, and so using the triangle inequality we see that

$$\begin{aligned} D(p_i, F) &\leq D(p_i, p_{i'}) + D(p_{i'}, F) \\ &= \bar{r}_i - r_{i'} + D(p_{i'}, F) \\ &\leq \bar{r}_i - \bar{r}_{i'} + (81s + 81) \cdot \bar{r}_{i'} \\ &\leq \bar{r}_i + (81s + 80) \cdot \bar{r}_{i'} \\ &\leq (81s + 81) \cdot \bar{r}_i \end{aligned}$$

If $r_i \leq 3\bar{r}_i$, then a more careful accounting is necessary. We know that, p_i is within distance $6r_i \cdot s$ of a node $p_j \in T_k$ (which may be p_i itself). Then, p_j either opens, or there exists a node $p_{j'}$ of a lower class such that $D(p_j, p_{j'}) \leq 2r_j$. In the former case $D(p_i, F) \leq 6r_i \cdot s$; in the latter case we have $D(p_i, p_{j'}) \leq D(p_i, p_j) + D(p_j, p_{j'}) \leq 6r_i s + 2r_j \leq (6s + 6) \cdot r_i$ since $r_j \leq 3r_i$ (owing to p_i and p_j being in the same class H_k).

So, within a distance $(6s + 6) \cdot r_i$ of p_i, there exists either an open node or a node of a lower class. In the latter case (in which there is a node $p_{j'}$ of a lower class), we repeat the above calculation for $p_{j'}$. Note that we do not know that $r_{j'} \leq 3\bar{r}_{j'}$, but we did not use that assumption about p_i in the preceding calculation.

We conclude then that within a distance $(6s+6) \cdot r_{j'}$ of $p_{j'}$, there exists either an open node or a node of a class $H_{k''}$, where $k'' \leq k - 2$. Since $p_{j'}$ is in a lower class than p_i, $r_{j'} < r_i$, and so it follows that, within a distance $(12s + 12) \cdot r_i$ of p_i, there exists either an open node or a node of a class of index less than or equal to $k - 2$.

Next, we repeat this analysis a *third* time and conclude that within a distance $(18s + 18) \cdot r_i$ of p_i, there exists either an open node or a node of a class of index less than or equal to $k - 3$.

Suppose now that there is no open node within distance $(18s + 18) \cdot r_i$ of p_i. Then there must exist a $p_{j'''} \in H_{k'''}$, where $k''' \leq k - 3$. Because class indices k''' and k are separated by at least two other classes, $9r_{j'''} \leq r_i$. We now apply the induction assumption.

$$D(p_{j'''}, F) \leq (81s + 81) \cdot \bar{r}_{j'''} \leq (81s + 81) \cdot r_{j'''} \leq (9s + 9) \cdot r_i$$

Putting the two cases together gives

$$\begin{aligned} D(p_i, F) &\leq \max\{(18s + 18) \cdot r_i, (18s + 18) \cdot r_i + D(p_{j'''}, F)\} \\ &\leq (18s + 18) \cdot r_i + (9s + 9) \cdot r_i \\ &= (27s + 27) \cdot r_i \end{aligned}$$

Since r_i was assumed to be less than or equal to $3\bar{r}_i$, we see that $D(p_i, F) \leq (27s + 27) \cdot r_i \leq (81s + 81) \cdot \bar{r}_i$, which completes the induction step.

Lemma 3. $\sum_{p_j \in F} \max\{0, r_j - D(p_j, p_i)\} \leq 9\bar{r}_i$

Proof. We begin by observing that we cannot simultaneously have $D(p_j, p_i) \leq r_j$ and $D(p_l, p_i) \leq r_l$ for $p_j, p_l \in F$ and $j \neq l$. Indeed, if this were the case, then $D(p_j, p_l) \leq r_j + r_l$. If p_j and p_l were in the same class H_x, then they would be adjacent in $G[H_x]$; this is impossible, for then they could not both be members of T_x (for a node in H_x, membership in T_x is necessary to join F). If p_j and p_l were in different classes, assume WLOG that $r_j < r_l$. Then $D(p_j, p_l) \leq r_j + r_l \leq 2r_l$, and p_l should not have opened. These contradictions imply that there is at most one open node p_j for which $D(p_j, p_i) \leq r_j$.

For the rest of this lemma, then, assume that $p_j \in F$ is the unique open node such that $D(p_j, p_i) \leq r_j$ (if such a p_j does not exist, there is nothing to prove). Note that p_i cannot be of a lower class than p_j (for else p_j would not have opened). Consequently, $r_j < 3r_i$.

Now, suppose $r_i \leq 3\bar{r}_i$. Then we have $r_j < 3r_i \leq 9\bar{r}_i$, and so the second component of $charge(p_i, F)$ is bounded by $9\bar{r}_i$, as desired.

If $r_i > 3\bar{r}_i$, then as in the previous lemma let $p_{i'}$ be a minimizer for $D(p_i, p_x) + r_x$, so that $\bar{r}_i = D(p_i, p_{i'}) + r_{i'}$. As before, $p_{i'}$ must be in a class of index strictly less than k. If it were true that $r_j \leq 9\bar{r}_i$, then we would be finished, so suppose this is not the case. We are thus in a case where $9\bar{r}_i < r_j < 3r_i$. Since $r_{i'} < \bar{r}_i$, we have $9r_{i'} < r_j$, and so we know that $p_{i'}$ must be in a lower class than p_j. Therefore, since $p_j \in F$, it must be true that $D(p_j, p_{i'}) \geq 2r_j \geq r_j$. So $r_j \leq D(p_j, p_i) + D(p_i, p_{i'})$, and

$$r_j - D(p_j, p_i) \leq D(p_i, p_{i'}) \leq \bar{r}_i \leq 9\bar{r}_i$$

which completes the proof of the lemma in the case that $r_i > 3\bar{r}_i$.

Theorem 2. *Algorithm 3 computes an MIS L of $C'[M]$ in $\frac{e[M]}{n} + \mathcal{O}(1)$ rounds.*

Proof. Each node p_i reserves a distinct range $\{D_i, D_i + 1, \ldots, D_i + d_i - 1\}$ of size d_i for labeling the d_i outgoing edges incident on it (Steps 1-3). This implies that the edges in $E[M]$ get unique labels in the range $\{0, 1, \ldots, e[M] - 1\}$. Sending each edge e to a node p_j with rank $\ell(e)(\mod n)$ means that each node receives at most $e[M]/n + 1$ edges (Step 4). Note that Steps 1-4 take at most one round each. Step 5 takes $e[M]/n + 1$ rounds, as this is the maximum number of edges received by a node in Step 4.

Lemma 4. *For any $d \geq 0$, the smallest k for which $t(k) \geq d$ is $\mathcal{O}(\log^* d)$.*

Proof. We introduce two functions. Let $tower(c, k)$ be defined recursively as $tower(c, 0) = 1$, and $tower(c, k) = c^{tower(c,k-1)}$ for $k > 0$. For $k \geq 2$, let

$$g(k) = e^{e^{tower((e/2),k-2)}}$$

Notice that for any $d > 0$, the smallest k for which $g(k) \geq d$ is $\mathcal{O}(\log^* d)$. We prove Lemma 4 by comparing $t(k)$ and $g(k)$. Specifically, we will show for any $k \geq 6$, $t(k) \geq g(k)$.

To prove this, we use induction on k. First, consider the base case where $k = 6$. Here,

$$t(6) = e^{e^{(1/2)e^{(1/2)e^{(1/2)e^{(1/2)e^{(1/2)}}}}}} > 53883$$

Also,

$$g(6) = e^{e^{tower((e/2),4)}} < 166$$

Therefore the claim holds for $k = 6$.

For the induction step, notice that

$$\begin{aligned} t(k+1) = e^{\sqrt{t(k)}} &\geq e^{\sqrt{g(k)}} \text{ (by induction hypothesis)} \\ &= e^{e^{(1/2)e^{tower((e/2),k-2)}}} \\ &\geq e^{e^{(e/2)^{tower((e/2),k-2)}}} = g(k+1) \end{aligned}$$

Therefore, our claim holds.

Lemma 5. *Consider an Iteration in Phase k, Stage i. Let $d = n^{1/2^i}$. The maximum degree of a node during this Iteration is d^2. Furthermore, the size of $S(d)$ is at most $n/t(k)$.*

Proof. Stage i does not begin until Stage $i-1$ has ended and all nodes in the graph have degree less than $n^{1/2^{i-1}} = d^2$. Phase k does not begin until Phase $k-1$ has ended and $|S(d)| \leq \frac{n}{t(k-1+1)} = \frac{n}{t(k)}$.

Lemma 6. *Algorithm 4 computes a 2-ruling set of C'.*

Proof. At any point, the only nodes removed from P are those in $M \cup N(M)$. Since we compute an MIS L of $C'[M]$ and include only these nodes in the final output T, every node in M is at distance at most one from a node in T and every node in $N(M)$ is at distance at most 2 from some node in T. Furthermore, no node left in P is a neighbor of any node in M and therefore no node that will be added to T in the future will cause any adjacencies with nodes added thus far to T. Thus T remains an independent set. When the algorithm terminates, all nodes in C' were either in M or in $N(M)$ at some point in the algorithm, and therefore T is a 2-ruling set of C'.

Lemma 7. *In any Iteration, the expected number of edges in the subgraph induced by M is bounded above by $2n$.*

Proof. Suppose that the Iteration under consideration is in Phase k, Stage i, with $d = n^{1/2^i}$. We will consider two types of edges: (i) edges between nodes in $\mathcal{S}(d)$, and (ii) all other edges. By Lemma 5, $|S(d)| \leq \frac{n}{t(k)}$, and $\max_{p \in S(d)} deg(p) \leq d^2$. Therefore, there are at most $\frac{nd^2}{t(k)}$ edges between nodes in $S(d)$, each of which is included in $C'[M]$ with probability at most $\frac{\sqrt{t(k)}}{d} \cdot \frac{\sqrt{t(k)}}{d} = \frac{t(k)}{d^2}$. Thus, the

expected number of edges in $C'[M]$ between nodes in $S(d)$ is at most $\frac{nd^2}{t(k)} \cdot \frac{t(k)}{d^2} = n$.

Next, consider edges incident upon nodes outside $S(d)$. There are at most n such nodes, each of which has degree at most d, and therefore there are at most nd edges incident upon nodes not in $S(d)$. Now consider an edge $\{p_i, p_j\}$, with $p_i \notin S(d)$. Node p_i is included in M with probability $1/\sqrt{d}$ and p_j is included in M with probability at most $1/\sqrt{d}$ (note that here we use the fact that the probability q of a node $p_j \in S(d)$ being included in M is at most $1/\sqrt{d}$). Therefore, each edge $\{p_i, p_j\}$ is included in $C'[M]$ with probability at most $\frac{1}{\sqrt{d}} \cdot \frac{1}{\sqrt{d}} = \frac{1}{d}$. This implies that the expected number of edges in $C'[M]$ incident upon nodes in $P \setminus S(d)$ is at most $nd \cdot \frac{1}{d} = n$.

Thus, in any Iteration the expected number of edges in the subgraph induced by M is bounded by $2n$.

Lemma 8. *Fix a Stage i, and suppose that $t(k+1) \leq d$. Then the expected number of Iterations (Lines 7-13) in Phase k before $|\mathcal{S}(d)| \leq \frac{n}{t(k+1)}$ is $\mathcal{O}(1)$.*

Proof. The probability that a node $p \in S(d)$ is deleted from $S(d)$ after one Iteration is at least

$$Pr[N(p) \cap M \neq \emptyset] = 1 - \prod_{p' \in N(p)} Pr[p' \notin M] \geq 1 - \left(1 - \frac{\sqrt{t(k)}}{d}\right)^d \geq 1 - e^{-\sqrt{t(k)}}$$

Thus the probability that p remains in $S(d)$ after an Iteration (in Phase k) is at most $e^{-\sqrt{t(k)}} = \frac{1}{t(k+1)}$. Therefore the expected size of $S(d)$ after one Iteration is at most $\frac{n}{t(k)} \cdot \frac{1}{t(k+1)} \leq \frac{n}{t(k+1)}$. This implies that the expected number of Iterations before $|S(d)| \leq \frac{n}{t(k+1)}$ is a constant.

Lemma 9. *Fix a Stage i, and suppose $t(k) \leq d < t(k+1)$. At the end of Phase $k+1$, there are at most $\mathcal{O}(n)$ edges in $C'[S(d)]$.*

Proof. Since $t(k) \leq d$, at the end of Phase k we have $|S(d)| \leq n/t(k+1)$. Also, prior to exiting Phase k, we set $q = \sqrt{t(k+1)}/d$. However, since this value of q is greater than $1/\sqrt{d}$, q is truncated at $1/\sqrt{d}$ and *lastPhase* is set. One more Phase, Phase $k+1$, is then executed, during which $q = 1/\sqrt{d}$.

With $q = 1/\sqrt{d}$, the probability of a node in $S(d)$ remaining in $S(d)$ after one Iteration is at most $e^{-\sqrt{d}}$. Therefore, the expected number of Iterations necessary before Phase $k+1$ terminates is $\mathcal{O}(1)$. When Phase $k+1$ does exit, we must have $|S(d)| \leq n \cdot e^{-\sqrt{d}}$. Now since the maximum degree of $S(d)$ is less than d^2, the number of edges in $C'[S(d)]$ is bounded by $\frac{n}{e^{\sqrt{d}}} \cdot d^2 = \mathcal{O}(n)$.

Theorem 3. *Algorithm 4 computes a 2-ruling set on C' in $\mathcal{O}(\log\log n \cdot \log^* n)$ rounds.*

Proof. In Lemma 6 we have shown that the output T is a 2-ruling set of C'.

To see the bound on the running time, first note that there are 3 calls to Algorithm 3 in Algorithm 4, namely at Lines 10, 18, and 22. The call to Algorithm 3 in Line 10 takes expected $\mathcal{O}(1)$ rounds because Lemma 7 shows that $C'[M]$ has at most $2n$ edges in expectation and by Theorem 2, Algorithm 3 takes 2 rounds to complete. The call to Algorithm 3 in Line 18 takes $\mathcal{O}(1)$ rounds because Lemma 9 establishes that $C'[M]$ has $\mathcal{O}(n)$ edges. Finally, the call to Algorithm 3 in Line 22 takes at most 10 rounds because every node in C' has degree at most 10 at this point in the algorithm. Thus every call to Algorithm 3 completes in $\mathcal{O}(1)$ rounds.

The upper bound on the running time now follows from the fact that there are $\mathcal{O}(\log \log n)$ Stages in the algorithm, with each Stage consisting of $\mathcal{O}(\log^* n)$ Phases, and each Phase consisting of $\mathcal{O}(1)$ Iterations in expectation.