A Super-Fast Distributed Algorithm for Bipartite Metric Facility Location

James Hegeman and Sriram V. Pemmaraju *

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

Abstract. The *facility location* problem consists of a set of *facilities* $\mathcal{F}$, a set of *clients* $\mathcal{C}$, 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 of facilities to *open*, and to connect each client to an open facility, so as to minimize the total facility opening costs plus connection costs. This paper presents the first expected-sub-logarithmic-round distributed $O(1)$-approximation algorithm in the $\mathcal{CONGEST}$ model for the *metric* facility location problem on the complete bipartite network with parts $\mathcal{F}$ and $\mathcal{C}$. Our algorithm has an expected running time of $O((\log\log n)^3)$ rounds, where $n = |\mathcal{F}| + |\mathcal{C}|$. This result can be viewed as a continuation of our recent work (ICALP 2012) in which we presented the first sub-logarithmic-round distributed $O(1)$-approximation algorithm for metric facility location on a *clique* network. The bipartite setting presents several new challenges not present in the problem on a clique network. We present two new techniques to overcome these challenges.

1 Introduction

This paper continues the recently-initiated exploration [2,1,7,9,16] of the design of sub-logarithmic, or "super-fast" distributed algorithms in low-diameter, bandwidth-constrained settings. To understand the main themes of this exploration, suppose that we want to design a distributed algorithm for a problem on a low-diameter network (we have in mind a clique network or a diameter-2 network). In one sense, this is a trivial task since the entire input could be shipped off to a single node in a single round and that node can simply solve the problem locally. On the other hand, the problem could be quite challenging if we were to impose reasonable constraints on bandwidth that prevent the fast delivery of the entire input to a small number of nodes. A natural example of this phenomenon is provided by the *minimum spanning tree* (MST) problem. Consider a clique network in which each edge (u, v) has an associated weight $w(u, v)$ of which only

* This work is supported in part by National Science Foundation grant CCF 0915543. Sriram V. Pemmaraju is the contact author. e-mail: sriram-pemmaraju@uiowa.edu, phone: (319) 353 2956.

the nodes u and v are aware. The problem is for the nodes to compute an MST of the edge-weighted clique such that after the computation, each node knows all MST edges. It is important to note that the problem is defined by $\Theta(n^2)$ pieces of input and it would take $\Omega\left(\frac{n}{B}\right)$ rounds of communication for all of this information to reach a single node (where B is the number of bits that can travel across an edge in each round). Typically, $B = O(\log n)$, and this approach is clearly too slow given our goal of completing the computation in a sub-logarithmic number of rounds. Lotker et al. [9] showed that the MST problem on a clique can in fact be solved in $O(\log \log n)$ rounds in the $\mathcal{CONGEST}$ model of distributed computation, which is a synchronous, message-passing model in which each node can send a message of size $O(\log n)$ bits to each neighbor in each round. The algorithm of Lotker et al. employs a clever merging procedure that, roughly speaking, causes the sizes of the MST components to square with each iteration, leading to an $O(\log \log n)$-round computation time. The overall challenge in this area is to establish the round complexity of a variety of problems that make sense in low-diameter settings. The area is largely open with few upper bounds and no non-trivial lower bounds known. For example, it has been proved that computing an MST requires $\Omega\left((\frac{n}{\log n})^{1/4}\right)$ rounds in the $\mathcal{CONGEST}$ model for diameter-3 graphs [10], but no lower bounds are known for diameter-2 or diameter-1 (clique) networks.

The focus of this paper is the *distributed facility location* problem, which has been considered by a number of researchers [12,4,14,15,2,1] 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* $\mathcal{F} = \{x_1, x_2, \ldots, x_{n_f}\}$, a set of *clients* $\mathcal{C} = \{y_1, y_2, \ldots, y_{n_c}\}$, a (nonnegative) *opening cost* f_i associated with each facility x_i, and a (nonnegative) *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 total facility opening costs 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 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:" 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. There are several constant-factor approximation algorithms for metric facility location (see [8] for a recent example). This approximation factor is known to be near-optimal [5].

More recently, the facility location problem has also been used as an abstraction for the problem of locating resources in wireless networks [3,13]. Motivated by this application, several researchers have considered the facility location problem in a distributed setting. In [12,14,15], as well as in the present work, the underlying communication network is a complete bipartite graph $G = \mathcal{F} + \mathcal{C}$, with $\mathcal{F}$ and $\mathcal{C}$ forming the bipartition. At the beginning of the algorithm, each

node, whether a facility or client, has knowledge of the connection costs ("distances") between itself and all nodes in the other part. In addition, the facilities know their opening costs. The problem is to design a distributed algorithm that runs on G in the $\mathcal{CONGEST}$ model and produces a subset $F \subseteq \mathcal{F}$ of facilities to *open*. To simplify exposition we assume that every cost in the problem input can be represented in $O(\log n)$ bits, thus allowing each cost to be transmitted in a single message. Each chosen facility will then open and provide services to any and all clients that wish to connect to it (each client must be served by some facility). The objective is to guarantee that $FacLoc(F) \leq \alpha \cdot OPT$, where OPT is the cost of an optimal solution to the given instance of facility location and α is a constant. We call this the BipartiteFacLoc problem. In this paper we present the first sub-logarithmic-round algorithm for the BipartiteFacLoc problem; specifically, our algorithm runs in $O((\log\log n_f)^2 \cdot \log\log \min\{n_f, n_c\})$ rounds in expectation, where $n_f = |\mathcal{F}|$ and $n_c = |\mathcal{C}|$. All previous distributed approximation algorithms for BipartiteFacLoc require a logarithmic number of rounds to achieve near-optimal approximation factors.

1.1 Overview of Technical Contributions

In a recent paper (ICALP 2012, [2]; full version available as [1]), we presented an expected-$O(\log\log n)$-round algorithm in the $\mathcal{CONGEST}$ model for CliqueFacLoc, the "clique version" of BipartiteFacLoc. The underlying communication network for this version of the problem is a clique with each edge (u, v) having an associated (connection) cost $c(u, v)$ of which only nodes u and v are aware (initially). Each node u also has an opening cost f_u, and may choose to open as a facility; nodes that do not open must connect to an open facility. The cost of the solution is defined as before – as the sum of the facility opening costs and the costs of established connections. Under the assumption that the connection costs form a metric, our algorithm for CliqueFacLoc yields an $O(1)$-approximation. We had hoped that a "super-fast" algorithm for BipartiteFacLoc would be obtained in a straightforward manner by extending our CliqueFacLoc algorithm. However, it turns out that moving from a clique communication network to a complete bipartite communication network raises several new and significant challenges related to information dissemination and a lack of adequate knowledge. Below we outline these challenges and our solutions to them.

Overview of solution to CliqueFacLoc. To solve CliqueFacLoc on an edge-weighted clique G [2,1] we reduce it to the problem of computing a 2-ruling set in an appropriately-defined spanning subgraph of G. A *β-ruling set* of a graph is an independent set S such that every node in the graph is at most β hops away from some node in S; a *maximal independent set* (MIS) is simply a 1-ruling set. The spanning subgraph H on which we compute a 2-ruling set is induced by clique edges whose costs are no greater than a pre-computed quantity which depends on the two endpoints of the edge in question.

We solve the 2-ruling set problem on the spanning subgraph H via a combination of deterministic and randomized sparsification. Briefly, each node selects

itself with a uniform probability p chosen such that the subgraph H' of H induced by the selected nodes has $\Theta(n)$ edges in expectation. The probability p is a function of n and the number of edges in H. We next deliver all of H' to every node. It can be shown that a graph with $O(n)$ edges can be completely delivered to every node in $O(1)$ rounds on a clique and since H' has $O(n)$ edges in expectation, the delivery of H' takes expected-$O(1)$ rounds. Once H' has been disseminated in this manner, each node uses the same (deterministic) rule to locally compute an MIS of H'. Following the computation of an MIS of H', nodes in the MIS and nodes in their 2-neighborhood are all deleted from H and H shrinks in size. Since H is now smaller, a larger probability p can be used for the next iteration. This increasing sequence of values for p results in a doubly-exponential rate of progress, which leads to an expected-$O(\log \log n)$-round algorithm for computing a 2-ruling set of H. See [1] for more details.

Challenges for BipartiteFacLoc. The same algorithmic framework can be applied to BipartiteFacLoc; however, challenges arise in trying to implement the ruling-set computation on a bipartite communication network. As in CliqueFacLoc [1], we define a particular graph H on the set of facilities with edges connecting pairs of facilities whose connection cost is bounded above. Note that there is no explicit notion of connection cost between facilities, but we use a natural extension of the facility-client connection costs $D(\cdot, \cdot)$ and define for each $x_i, x_j \in \mathcal{F}$, $D(x_i, x_j) := \min_{y \in \mathcal{C}} D(x_i, y) + D(x_j, y)$. The main algorithmic step now is to compute a 2-ruling set on the graph H. However, difficulties arise because H is not a subgraph of the communication network G, as it was in the CliqueFacLoc setting. In fact, initially a facility x_i does not even know to which other facilities it is adjacent to in H. This adjacency knowledge is collectively available only to the clients. A client y *witnesses* edge $\{x_i, x_j\}$ in H if $D(x_i, y) + D(x_j, y)$ is bounded above by a pre-computed quantity associated with the facility-pair x_i, x_j. However, (initially) an individual client y cannot certify the *non-existence* of any potential edge between two facilities in H; as, unbeknownst to y, some other client may be a witness to that edge. Furthermore, the same edge $\{x_i, x_j\}$ could have many client-witnesses. This "affirmative-only" adjacency knowledge and the duplication of this knowledge turn out to be key obstacles to overcome. For example, in this setting, it seems difficult to even figure out how many edges H has.

Thus, an example of a problem we need to solve is this: without knowing the number of edges in H, how do we correctly pick a probability p that will induce a random subgraph H' with $\Theta(n)$ edges? Duplication of knowledge of H leads to another problem as well. Suppose we did manage to pick a "correct" value of p and have induced a subgraph H' having $\Theta(n)$ edges. In the solution to CliqueFacLoc, we were able to deliver all of H' to a single node (in fact, to every node). In the bipartite setting, how do we deliver H' to a single node given that even though it has $O(n)$ edges, information duplication can cause the sum of the number of adjacencies witnessed by the clients to be as high as $\Omega(n^2)$?

We introduce new techniques to solve each of these problems. These techniques are sketched below.

– **Message dissemination with duplicates.** We model the problem of delivering all of H' to a single node as the following message-dissemination problem on a complete bipartite graph.

> **Message Dissemination with Duplicates (MDD).**
> Given a bipartite graph $G = \mathcal{F} + \mathcal{C}$, with $n_f := |\mathcal{F}|$ and $n_c := |\mathcal{C}|$, suppose that there are n_f messages that we wish to be known to all client nodes in $\mathcal{C}$. Initially, each client possesses some subset of the n_f messages, with each message being possessed by at least one client. Suppose, though, that no client y_j has any information about which of its messages are also held by any other client. Disseminate all n_f messages to each client in the network in expected-sub-logarithmic time.

  We solve this problem by presenting an algorithm that utilizes probabilistic hashing to iteratively reduce the number of duplicate copies of each message. Note that if no message exists in duplicate, then the total number of messages held is only n_f, and each can be sent to a distinct facility which can then broadcast it to every client. The challenge, then, lies in coordinating bandwidth usage so as to avoid "bottlenecks" that could be caused by message duplication. Our algorithm for MDD runs in $O(\log\log\min\{n_f, n_c\})$ rounds in expectation.

– **Random walk over a probability space.** Given the difficulty of quickly acquiring even basic information about H (e.g., how many edges does it have?), we have no way of setting the value of p correctly. So we design an algorithm that performs a random walk over a space of $O(\log\log n_f)$ probabilities. The algorithm picks a probability p, uses this to induce a random subgraph H' of H, and attempts to disseminate H' to all clients within $O(\log\log\min\{n_f, n_c\})$ rounds. If this dissemination succeeds, p is modified in one way (increased appropriately), otherwise p is modified differently (decreased appropriately). This technique can be modeled as a random walk on a probability space consisting of $O(\log\log n_f)$ elements, where the elements are distinct values that p can take. We show that after a random walk of length at most $O(\log\log n_f)$, sufficiently many edges of H are removed, leading to $O(\log\log n_f)$ levels of progress. Thus we have a total of $O((\log\log n_f)^2)$ steps and since in each step an instance of MDD is solved for disseminating adjacencies, we obtain an expected-$O((\log\log n_f)^2 \cdot \log\log\min\{n_f, n_c\})$-round algorithm for computing a 2-ruling set of H.

To summarize, our paper makes three main technical contributions. (i) We show (in Section 2) that the framework developed in [1] to solve CliqueFacLoc can be used, with appropriate modifications, to solve BipartiteFacLoc. Via this algorithmic framework, we reduce BipartiteFacLoc to the problem of computing a 2-ruling set of a graph induced by facilities in a certain way. (ii) In order to compute a 2-ruling set of a graph, we need to disseminate graph adjacencies whose knowledge is distributed among the clients with possible duplication.

We model this as a message dissemination problem and show (in Section 3), using a probabilistic hashing scheme, how to efficiently solve this problem on a complete bipartite graph. (iii) Finally, we present (in Section 4) an algorithm that performs a random walk over a probability space to efficiently compute a 2-ruling set of a graph, without even basic information about the graph. This algorithm repeatedly utilizes the procedure for solving the message-dissemination problem mentioned above.
Note: This paper does not contain any proofs, due to space restrictions. All proofs appear in the archieved full version of the paper [6].

2 Reduction to the Ruling Set Problem

In this section we reduce BipartiteFacLoc to the ruling set problem on a certain graph induced by facilities. The reduction is achieved via the distributed facility location algorithm called LocateFacilities and shown as Algorithm 1. This algorithm is complete except that it calls a subroutine, RulingSet(H, s) (in Step 4), to compute an s-ruling set of a certain graph H induced by facilities. In this section we first describe Algorithm 1 and then present its analysis. It is easily observed that all the steps in Algorithm 1, except the one that calls RulingSet(H, s) take a total of $O(1)$ communication rounds. Thus the running time of RulingSet(H, s) essentially determines the running time of Algorithm 1. Furthermore, we show that if F^* is the subset of facilities opened by Algorithm 1, then $FacLoc(F^*) = O(s) \cdot OPT$. In the remaining sections of the paper we show how to implement RulingSet$(H, 2)$ in expected $O((\log\log n_f)^2 \cdot \log\log\min\{n_f, n_c\})$ rounds. This yields an expected $O((\log\log n_f)^2 \cdot \log\log\min\{n_f, n_c\})$-round, $O(1)$-approximation algorithm for BipartiteFacLoc.

2.1 Algorithm

Given $\mathcal{F}$, $\mathcal{C}$, $D(\cdot,\cdot)$, and $\{f_i\}$, define the *characteristic radius* r_i of facility x_i to be the nonnegative real number satisfying $\sum_{y \in B(x_i, r_i)}(r_i - D(x_i, y)) = f_i$, where $B(x, r)$ (the *ball* of radius r) denotes the set of clients y such that $D(x, y) \leq r$. This notion of a characteristic radius was first introduced by Mettu and Plaxton [11], who use it to drive their sequential, greedy algorithm. We extend the client-facility distance function $D(\cdot,\cdot)$ to facility-facility distances; let $D : \mathcal{F} \times \mathcal{F} \to \mathbb{R}^+ \cup \{0\}$ be defined by $D(x_i, x_j) = \min_{y_k \in \mathcal{C}}\{D(x_i, y_k) + D(x_j, y_k)\}$. With these definitions in place we are ready to describe Algorithm 1. The algorithm consists of three stages, which we now describe.

Stage 1. (Steps 1-2) Each facility knows its own opening cost and the distances to all clients. So in Step 1 facility x_i computes r_i and broadcasts that value to all clients. Once this broadcast is complete, each client knows all of the r_i values. This enables every client to compute the same partition of the facilities into classes as follows (Step 2). Define the special value $r_0 := \min_{1 \leq i \leq n_f}\{r_i\}$. Define the class V_k, for $k = 0, 1, \ldots$, to be the set of facilities x_i such that

Algorithm 1 LocateFacilities

Input: A complete bipartite graph G with partition $(\mathcal{F}, \mathcal{C})$; (bipartite) metric $D(\cdot,\cdot)$; opening costs $\{f_i\}_{i=1}^{n_f}$; a sparsity parameter $s \in \mathbb{Z}^+$
Assumption: Each facility knows its own opening cost and its distances to all clients; each client knows its distances to all facilities
Output: A subset of facilities (a *configuration*) to be declared open.

1. Each facility x_i computes and broadcasts its radius r_i to all clients; $r_0 := \min_i r_i$.
2. Each client computes a partition of the facilities into classes $\{V_k\}$ such that $3^k \cdot r_0 \leq r_i < 3^{k+1} \cdot r_0$ for $x_i \in V_k$.
3. For $k = 0, 1, \ldots$, define a graph H_k with vertex set V_k and edge set:
   $\{\{x_i, x_{i'}\} \mid x_i, x_{i'} \in V_k \text{ and } D(x_i, x_{i'}) \leq r_i + r_{i'}\}$
   (Observe from the definition of facility distance that such edges may be known to as few as one client, or as many as all of them.)
4. All nodes in the network use procedure RulingSet($\cup_k H_k, s$) to compute a 2-ruling set T of $\cup_k H_k$. T is known to every client. We use T_k to denote $T \cap V_k$.
5. Each client y_j sends an `open` message to each facility x_i, if and only if both of the following conditions hold:
   (i) x_i is a member of the set $T_k \subseteq V_k$, for some k.
   (ii) y_j is not a witness to the existence of a facility $x_{i'}$ belonging to a class $V_{k'}$, with $k' < k$, such that $D(x_i, x_{i'}) \leq 2r_i$.
6. Each facility x_i opens, and broadcasts its status as such, if and only if x_i received an `open` message from every client.
7. Each client connects to the nearest open facility.

$3^k \cdot r_0 \leq r_i < 3^{k+1} \cdot r_0$. Every client computes the class into which each facility in the network falls.

Stage 2. (Steps 3-4) Now that the facilities are divided into classes having comparable r_i's, and every client knows which facility is in each class, we focus our attention on class V_k. Suppose $x_i, x_{i'} \in V_k$. Then we define x_i and $x_{i'}$ to be *adjacent* in class V_k if $D(x_i, x_{i'}) \leq r_i + r_{i'}$ (Step 3). These adjacencies define the graph H_k with vertex set V_k. Note that two facilities x_i, $x_{i'}$ in class V_k are adjacent if and only if there is at least one client *witness* for this adjacency. Next, the network computes an s-ruling set T of $\cup_k H_k$ with procedure RulingSet() (Step 4). We describe a super-fast implementation of RulingSet() in Section 4. After a ruling set T has been constructed, every client knows all the members of T. Since the H_k's are disjoint, $T_k := T \cap V_k$ is a 2-ruling set of H_k for each k.

Stage 3. (Steps 5-7) Finally, a client y_j sends an `open` message to facility x_i in class V_k if (i) $x_i \in T_k$, and (ii) there is no facility $x_{i'}$ of class $V_{k'}$ such that $D(x_i, y_j) + D(x_{i'}, y_j) \leq 2r_i$, and for which $k' < k$ (Step 5). A facility opens if it receives `open` messages from all clients (Step 6). Lastly, open facilities declare themselves as such in a broadcast, and every client connects to the nearest open facility (Step 7).

2.2 Analysis

The approximation-factor analysis of Algorithm 1 is similar to the analysis of our algorithm for CliqueFacLoc [1]. Here we present a brief summary.
First we show a lower bound on the cost *any* solution to BipartiteFacLoc. For $y_j \in \mathcal{C}$, define $\overline{r}_j$ as $\overline{r}_j = \min_{1 \le i \le n_f}\{r_i + D(x_i, y_j)\}$. See [1] for a motivation for this definition.

Lemma 1. *$FacLoc(F) \ge (\sum_{j=1}^{n_c} \overline{r}_j)/6$ for any subset $F \subseteq \mathcal{F}$.*

To obtain an upper bound on the cost of the solution produced by Algorithm 1 we start by "charging" the cost of a facility location solution to clients in a standard way [11]. For a client $y_j \in \mathcal{C}$ and a facility subset F, define the *charge* of y_j with respect to F by $charge(y_j, F) = D(F, y_j) + \sum_{x_i \in F} \max\{0, r_i - D(x_i, y_j)\}$. Simple algebraic manipulation can be used to show that for any facility subset F, $FacLoc(F)$ is equal to $\sum_{j=1}^{n_c} charge(y_j, F)$. Finally, if F^* is the subset of facilities selected by Algorithm 1, we show the following upper bounds.

Lemma 2. $D(F^*, y_j) \le (s+1) \cdot 15 \cdot \overline{r}_j$.

Lemma 3. $\sum_{x_i \in F^*} \max\{0, r_i - D(x_i, y_j\} \le 3 \cdot \overline{r}_j$.

Putting the lower bound, charging scheme, and upper bound together gives $FacLoc(F^*) \le (15s+33) \cdot \sum_{j=1}^{n_c} \overline{r}_j \le 6 \cdot (15s+33) \cdot OPT$. Also, noting that all the steps in Algorithm 1, except the one that calls RulingSet$(\cup_k H_k, s)$ take a total of $O(1)$ communication rounds, we obtain the following theorem.

Theorem 1. *Algorithm 1 (*LocateFacilities*) computes an $O(s)$-factor approximation to* BipartiteFacLoc *in $O(\mathcal{T}(n,s))$ rounds, where $\mathcal{T}(n,s)$ is the running time of procedure* RulingSet*(H, s), called an n-node graph H.*

3 Dissemination on a Bipartite Network

In the previous section we reduced BipartiteFacLoc to the problem of computing an s-ruling set on a graph $H = \cup_k H_k$ defined on facilities. Our technique for finding an s-ruling set involves selecting a set M of facilities at random, disseminating the induced subgraph $H[M]$ to every client and then having each client locally compute an MIS of $H[M]$ (details appear in Section 4). A key subroutine needed to implement this technique is one that can disseminate $H[M]$ to every client efficiently, provided the number of edges in $H[M]$ is at most n_f. In Section 1 we abstracted this problem as the **Message Dissemination with Duplicates** (MDD) problem. In this section, we present a randomized algorithm for MDD that runs in expected $O(\log\log\min\{n_f, n_c\})$ communication rounds.

Recall that the difficulty in disseminating $H[M]$ is the fact that the adjacencies in this graph are witnessed only by clients, with each adjacency being witnessed by at least one client. However, an adjacency can be witnessed by many clients and a client is unaware of who else has knowledge of any particular edge.

Thus, even if $H[M]$ has at most n_f edges, the total number of adjacency observations by the clients could be as large as n_f^2. Below we use iterative probabilistic hashing to rapidly reduce the number of "duplicate" witnesses to adjacencies in $H[M]$. Once the total number of distinct adjacency observations falls to $48n_f$, it takes only a constant number of additional communication rounds for the algorithm to finish disseminating $H[M]$. The constant "48" falls out easily from our analysis (Lemma 7, in particular) and we have made no attempt to optimize it in any way.

3.1 Algorithm

The algorithm proceeds in iterations and in each iteration a hash function is chosen at random for hashing messages held by clients onto facilities. Denote the universe of possible adjacency messages by $\mathcal{U}$. Since messages represent adjacencies among facilities, $|\mathcal{U}| = \binom{n_f}{2}$. However, it is convenient for $|\mathcal{U}|$ to be equal to n_f^2 and so we extend $\mathcal{U}$ by dummy messages so that this is the case. We now define a family $\mathcal{H}_\mathcal{U}$ of hash functions from $\mathcal{U}$ to $\{1, 2, \ldots, n_f\}$ and show how to pick a function from this family, uniformly at random. To define $\mathcal{H}_\mathcal{U}$, fix an ordering $m_1, m_2, m_3, \ldots$ of the messages of $\mathcal{U}$. Partition $\mathcal{U}$ into groups of size n_f, with messages $m_1, m_2, \ldots, m_{n_f}$ as the first group, the next n_f elements as the second group, and so on. The family $\mathcal{H}_\mathcal{U}$ is obtained by independently mapping each group of messages onto $(1, 2, \ldots, n_f)$ via a cyclic permutation. For each group of n_f messages in $\mathcal{U}$, there are precisely n_f such cyclic maps for it, and so a map in $\mathcal{H}_\mathcal{U}$ can be selected uniformly at random by having each facility choose a random integer in $\{1, 2, \ldots, n_f\}$ and broadcast this choice to all clients (in the first round of an iteration). Each client then interprets the integer received from facility x_i as the image of message $m_{(i-1)\cdot n_f+1}$.

In round 2, each client chooses a destination facility for each adjacency message in its possession (note that no client possesses more than n_f messages), based on the hash function chosen in round 1. For a message m in the possession of client y_j, y_j computes the hash $h(m)$ and marks m for delivery to facility $x_{h(m)}$. In the event that more than one of y_j's messages are intended for the same recipient, y_j chooses one uniformly at random for *correct* delivery, and marks the other such messages as "leftovers." During the communication phase of round 2, then, client y_j delivers as many messages as possible to their correct destinations; leftover messages are delivered uniformly at random over unused communication links to other facilities.

In round 3, a facility has received a collection of up to n_c messages, some of which may be duplicates of each other. After throwing away all but one copy of any duplicates received, each facility announces to client y_1 the number of (distinct) messages it has remaining. In round 4, client y_1 has received from each facility its number of distinct messages, and computes for each an index (modulo n_c) that allows facilities to coordinate their message transfers in the next round. Client y_1 transmits the indices back to the respective facilities in round 5.

In round 6, facilities transfer their messages back across the bipartition to the clients, beginning at their determined index (received from client y_1) and working modulo n_c. This guarantees that the numbers of messages received by two clients y_j, $y_{j'}$ in this round can differ by no more than one. (Although it is possible that some of these messages will "collapse" as duplicates.) Clients now possess subsets of the original n_f messages, and the next iteration can begin.

Algorithm 2 DisseminateAdjacencies

Input: A complete bipartite graph G, with partition $(\mathcal{F}, \mathcal{C})$; an overlay network H on $\mathcal{F}$ with $|E[H]| \leq n_f$
Assumption: For each adjacency e' in H, *one or more* clients has knowledge of e'
Output: Each client should know the entire contents of $E[H]$

1. **while** *true* **do**
   Start of Iteration:
2. Each client y_j sends the number of distinct messages currently held, n_j, to facility x_1.
3. **if** $\sum_{j=1}^{n_c} n_j \leq 48n_f$ **then**
4. Facility x_1 broadcasts a *break* message to each client.
5. Client y_1, upon receiving a *break* message, broadcasts a *break* message to each facility.
   end-if-then
6. Each facility x_i broadcasts an integer in $\{1, \ldots, n_f\}$ chosen uniformly at random; this collection of broadcasts determines a map $h \in \mathcal{H}_{\mathcal{U}}$.
7. For each adjacency message m' currently held, client y_j maps m' to $x_{h(m')}$.
8. For each $i \in \{1, \ldots, n_f\}$, if $|\{m' \text{ held by } y_j \ : \ h(m') = i\}| > 1$, client y_j chooses one message to send to x_i at random from this set and marks the others as *leftovers*.
9. Each client y_j sends the messages chosen in Lines 7-8 to their destinations; *leftover* messages are delivered to other facilities (for whom y_j has no intended message) in an arbitrary manner (such that y_j sends at most one message to each facility).
10. Each facility x_i receives a collection of at most n_c facility adjacency messages; if duplicate messages are received, x_i discards all but one of them so that the messages held by x_i are distinct.
11. Each facility x_i sends its number of distinct messages currently held, b_i, to client y_1.
12. Client y_1 responds to each facility x_i with an index $c(i) = (\sum_{k=1}^{i-1} b_k \mod n_c)$.
13. Each facility x_i distributes its current messages evenly to the clients in the set $\{y_{c(i)+1}, y_{c(i)+2}, \ldots, y_{c(i)+b_i}\}$ (where indexes are reduced modulo n_c as necessary).
14. Each client y_j receives at most n_f messages; the numbers of messages received by any two clients differ by at most one.
15. Each client discards any duplicate messages held.
    End of Iteration:
16. At this point, at most $48n_f$ total messages remain among the n_c clients; these messages may be distributed evenly to the facilities in $O(1)$ communication rounds.
17. The n_f facilities can now broadcast the (at most) $2n_f$ messages to all clients in $O(1)$ rounds.

3.2 Analysis

Algorithm 2 is proved correct by observing that (i) the algorithm terminates only when dissemination has been completed; and (ii) for a particular message m', in any iteration, there is a nonzero probability that all clients holding a copy of m' will deliver m' correctly, after which there will never be more than one copy of m' (until all messages are broadcast to all clients at the end of the algorithm). The running time analysis of Algorithm 2 starts with two lemmas that follow from our choice of the probabilistic hash function.

Lemma 4. *Suppose that, at the beginning of an iteration, client y_j possesses a collection S_j of messages, with $|S_j| = n_j$. Let $E_{i,j}$ be the event that at least one message in S_j hashes to facility x_i. Then the probability of $E_{i,j}$ (conditioned on all previous iterations) is bounded below by $1 - e^{-n_j/n_f}$.*

Lemma 5. *Suppose that, at the beginning of an iteration, client y_j possesses a collection S_j of messages, with $|S_j| = n_j$. Let $M_j \subseteq S_j$ be the subset of messages that are correctly delivered by client y_j in the present iteration. Then the expected value of $|M_j|$ (conditioned on previous iterations) is bounded below by $n_j - \frac{n_j^2}{2n_f}$.*

By Lemma 5, the number of incorrectly delivered messages in S_j is bounded above (in expectation) by $\frac{n_j^2}{2n_f}$. Informally speaking, this implies that the sequence $n_f, \frac{n_f}{2}, \frac{n_f}{2^3}, \frac{n_f}{2^7}, \ldots$ bounds from above the number of incorrectly delivered messages (in expectation) in each iteration. This doubly-exponential rate of decrease in the number of undelivered messages leads to the expected-doubly-logarithmic running time of the algorithm.

We now step out of the context of a single client and consider the progress of the algorithm on the whole. Using Lemma 5, we derive the following recurrence for the expected total number of messages held by all clients at the beginning of each iteration.

Lemma 6. *Suppose that the algorithm is at the beginning of iteration I, $I \geq 2$, and let T_I be the total number of messages held by all clients (i.e. $T_I = \sum_{j=1}^{n_c} n_j(I)$, where $n_j(I)$ is the number of messages held by client y_j at the beginning of iteration I). Then the conditional expectation of T_{I+1} given T_I, $\mathbf{E}(T_{I+1} \mid T_I)$, satisfies*

$$\mathbf{E}(T_{I+1} \mid T_I) \leq \begin{cases} n_f + \frac{(T_I + n_c)^2}{2n_f \cdot n_c} & \text{if } T_I > n_c \\ n_f + \frac{T_I}{2n_f} & \text{if } T_I \leq n_c \end{cases}$$

We now define a sequence of variables t_i (via the recurrence below) that bounds from above the expected behavior of the sequence of T_I's established in the

previous lemma. Let $t_1 = n_f \cdot \min\{n_f, n_c\}$, $t_i = \frac{1}{2}t_{i-1}$ for $2 \leq i \leq 5$, and for $i > 5$, define t_i by

$$t_i = \begin{cases} 2n_f + \frac{(t_{i-1}+n_c)^2}{n_f \cdot n_c} & \text{if } t_{i-1} > n_c \\ 2n_f + \frac{t_{i-1}}{n_f} & \text{if } t_{i-1} \leq n_c \end{cases}$$

The following lemma establishes that the t_i's fall rapidly.

Lemma 7. *The smallest index i for which $t_i \leq 48n_f$ is at most* $\log\log\min\{n_f, n_c\}+2$.

Lemma 8. *For $i > 5$, if $T_I \leq t_i$, then the conditional probability (given iterations 1 through $I-1$) of the event that $T_{I+1} \leq t_{i+1}$ is bounded below by $\frac{1}{2}$.*

Theorem 2. *Algorithm 2 solves the dissemination problem in $O(\log\log\min\{n_f, n_c\})$ rounds in expectation.*

4 Computing a 2-Ruling Set of Facilities

In this section, we show how to efficiently compute a 2-ruling set on the graph H (with vertex set $\mathcal{F}$) constructed in Algorithm 1 (LocateFacilities). Our algorithm (called Facility2RulingSet and described as Algorithm 3) computes a 2-ruling set in H by performing *iterations* of a procedure that combines randomized and deterministic sparsification steps. In each iteration, each facility chooses (independently) to join the *candidate set* M with probability p. Two neighbors in H may both have chosen to join M, so M may not be independent in H. We would therefore like to select an MIS of the graph induced by M, $H[M]$. In order to do this, the algorithm attempts to communicate all known adjacencies in $H[M]$ to every client in the network, so that each client may (deterministically) compute the same MIS. The algorithm relies on Algorithm DisseminateAdjacencies (Algorithm 2) developed in Section 3 to perform this communication.

For Algorithm DisseminateAdjacencies to terminate quickly, we require that the number of edges in $H[M]$ be $O(n_f)$. This requires the probability p to be chosen carefully as a function of n_f and the number of edges in H. Due to the lack of aggregated information, nodes of the network do not generally know the number of edges in H and thus the choice of p may be "incorrect" in certain iterations. To deal with the possibility that p may be too large (and hence $H[M]$ may have too many edges), the dissemination procedure is not allowed to run indefinitely – rather, it is cut off after $7\log\log\min\{n_f, n_c\}$ iterations of disseminating hashing. If dissemination was successful, i.e. the subroutine completed prior to the cutoff, then each client receives complete information about the adjacencies in $H[M]$, and thus each is able to compute the same MIS in $H[M]$. Also, if dissemination was successful, then M and its neighborhood, $N(M)$, are removed from H and the next iteration is run with a larger probability

Algorithm 3 FACILITY2RULINGSET

Input: Complete bipartite graph G with partition $(\mathcal{F}, \mathcal{C})$ and H, an overlay network on $\mathcal{F}$.
Output: A 2-ruling set T of H

1. $i := 1$; $p := p_1 = \frac{1}{8 \cdot n_f^{1/2}}$; $T := \emptyset$
2. **while** $|E(H)| > 0$ **do**
   Start of Iteration:
3. $M := \emptyset$
4. Each facility x joins M with a probability p.
5. Run Algorithm DISSEMINATEADJACENCIES for $7 \log\log \min\{n_f, n_c\}$ iterations to communicate the edges in $H[M]$ to all clients in the network.
6. **if** DISSEMINATEADJACENCIES completes in the allotted number of iterations **then**
7. Each client computes the same MIS L on M using a deterministic algorithm.
8. $T := T \cup L$
9. Remove $M \cup N(M)$ from H.
10. $i := i + 1$; $p := p_i = \frac{1}{8 \cdot n_f^{2^{-i}}}$
11. **else**
12. $i := i - 1$; $p := p_i = \frac{1}{8 \cdot n_f^{2^{-i}}}$
13. **if** $|E(H)| = 0$ **then break**;
    End of Iteration:
14. Output T.

p. On the other hand, if dissemination was unsuccessful, the current iteration of FACILITY2RULINGSET is terminated and the next iteration is run with a smaller probability p (to make success more likely the next time).

To analyze the progress of the algorithm, we define two notions – *states* and *levels*. For the remainder of this section, we use the term *state* (of the algorithm) to refer to the current probability value p. The probability p can take on values $\left(\frac{1}{8 \cdot n_f^{2^{-i}}}\right)$ for $i = 0, 1, \ldots, \Theta(\log\log n_f)$. We use the term *level* to refer to the progress made up until the current iteration. Specifically, the jth level L_j, for $j = 0, 1, \ldots, \Theta(\log\log n_f)$, is defined as having been reached when the number of facility adjacencies remaining in H becomes less than or equal to $l_j = 8 \cdot n_f^{1+2^{-j}}$. In addition, we define one special level L_* as the level in which no facility adjacencies remain. These values for the states and levels are chosen so that, once level L_i has been reached, one iteration run in state $i + 1$ has at least a probability-$\frac{1}{2}$ chance of advancing progress to level L_{i+1}.

4.1 Analysis

It is easy to verify that the set T computed by Algorithm 3 (FACILITY2RULINGSET) is a 2-ruling set and we now turn our attention to the expected running time of this algorithm. The algorithm halts exactly when level L_* is reached (this

termination condition is detected in Line 15), and so it suffices to bound the expected number of rounds necessary for progress (removal of edges from H) to reach level L_*. The following lemmas show that quick progress is made when the probability p matches the level of progress made thus far.

Lemma 9. *Suppose $|E(H)| \leq l_i$ (progress has reached level L_i) and in this situation one iteration is run in state $i+1$ (with $p = p_{i+1}$). Then in this iteration, the probability that Algorithm* DisseminateAdjacencies *succeeds is at least $\frac{3}{4}$.*

Lemma 10. *Suppose $|E(H)| \leq l_i$ (progress has reached level L_i). Then, after one iteration run in state $i+1$ (with $p = p_{i+1}$), the probability that level L_{i+1} will be reached (where $|E(H)| \leq l_{i+1}$) is at least $\frac{1}{2}$.*

Thus, once level L_i has been reached, we can expect that only a constant number of iterations run in state $i+1$ would be required to reach level L_{i+1}. Therefore, the question is, "How many iterations of the algorithm are required to execute state $i+1$ enough times?" To answer this question, we abstract the algorithm as a stochastic process that can be modeled as a (non-Markov) simple random walk on the integers $0, 1, 2, \ldots, \Theta(\log \log n_f)$ with the extra property that, whenever the random walk arrives at state $i+1$, a (fair) coin is flipped. We place a bound on the expected number of steps before this coin toss comes up heads.

First, consider the return time to state $i+1$. In order to prove that the expected number of iterations (steps) necessary before either $|E(H)| \leq l_{i+1}$ or $p = p_{i+1}$ is $O(\log \log n_f)$, we consider two regimes – $p > p_{i+1}$ and $p < p_{i+1}$. When p is large (in the regime consisting of probability states intended for fewer edges than currently remain in H), it is likely that a single iteration of Algorithm 3 will generate a large number of adjacencies between candidate facilities. Thus, dissemination will likely not complete before "timing out," and it is likely that p will be decreased prior to the next iteration. Conversely, when p is small (in the regime consisting of probability states intended for more edges than currently remain in H), a single iteration of Algorithm 3 will likely generate fewer than n_f adjacencies between candidate facilities, and thus it is likely that dissemination will complete before "timing out." In this case, p will advance prior to the next iteration. This analysis is accomplished in the following lemmas and leads to the subsequent theorem.

Lemma 11. *Consider a simple random walk on the integers $[0, i]$ with transition probabilities $\{p_{j,k}\}$ satisfying $p_{j,j+1} = \frac{3}{4}$ $(j = 0, \ldots, i-1)$, $p_{j,j-1} = \frac{1}{4}$, $(j = 1, \ldots, i)$, $p_{i,i} = \frac{3}{4}$, and $p_{0,0} = \frac{1}{4}$. For such a random walk beginning at 0, the expected hitting time of i is $O(i)$.*

Lemma 12. *When $j \leq i$, the expected number of iterations required before returning to state $i+1$ is $O(\log \log n_f)$.*

Lemma 13. *When $j > i$, the expected number of iterations required before returning to state $i+1$ or advancing to at least level L_{i+1} is $O(\log \log n_f)$.*

Lemma 14. *Suppose that Algorithm 3 has reached level L_i, and let T_{i+1} be a random variable representing the number of iterations necessary before reaching level L_{i+1}. Then $\mathbf{E}(T_{i+1}) = O(\log\log n_f)$.*

Theorem 3. *Algorithm 3 has an expected running time of $O((\log\log n_f)^2 \cdot \log\log\min\{n_f, n_c\})$ rounds in the $\mathcal{CONGEST}$ model.*

References

1. A. Berns, J. Hegeman, and S. V. Pemmaraju. Super-fast distributed algorithms for metric facility location. *CoRR*. Archived on Aug 11, 2013.
2. A. Berns, J. Hegeman, and S. V. Pemmaraju. Super-fast distributed algorithms for metric facility location. In *ICALP*, pages 428—439, 2012.
3. C. Frank. *Algorithms for Sensor and Ad Hoc Networks*. Springer, 2007.
4. J. Gehweiler, C. Lammersen, and C. Sohler. 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*, SPAA '06, pages 237–243, New York, NY, USA, 2006. ACM, ACM Press.
5. S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 649—657. Society for Industrial and Applied Mathematics, 1998.
6. J. Hegeman and S. V. Pemmaraju. A super-fast distributed algorithm for bipartite metric facility location. *CoRR*. Archived on Aug 12, 2013.
7. Christoph Lenzen. Optimal deterministic routing and sorting on the congested clique. *CoRR*, abs/1207.1852, 2012.
8. S. Li. A 1.488-approximation algorithm for the uncapacitated facility location problem. In *Proceedings of the 38th international colloquium on automata, languages and programming*, ICALP '11, pages 77—88, Berlin, Heidelberg, 2011. Springer-Verlag.
9. Z. Lotker, B. Patt-Shamir, E. Pavlov, and D. Peleg. Minimum-weight spanning tree construction in O(log log n) communication rounds. *SIAM J. Comput.*, 35(1):120—131, 2005.
10. Zvi Lotker, Boaz Patt-Shamir, and David Peleg. Distributed mst for constant diameter graphs. *Distributed Computing*, 18(6):453–460, 2006.
11. R. R. Mettu and C. G. Plaxton. The online median problem. *SIAM J. Comput.*, 32(3):816—832, 2003.
12. T. Moscibroda and R. Wattenhofer. Facility location: distributed approximation. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 108—117, New York, NY, USA, 2005. ACM, ACM Press.
13. S. Pandit and S. V. Pemmaraju. Finding facilities fast. *Distributed Computing and Networking*, pages 11—24, 2009.
14. S. Pandit and S. V. Pemmaraju. Return of the primal-dual: distributed metric facility location. In *Proceedings of the 28th ACM symposium on Principles of distributed computing*, PODC '09, pages 180—189, New York, NY, USA, 2009. ACM, ACM Press.
15. S. Pandit and S. V. Pemmaraju. Rapid randomized pruning for fast greedy distributed algorithms. In *Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 325—334. ACM, 2010.
16. B. Patt-Shamir and M. Teplitsky. The round complexity of distributed sorting: extended abstract. In *PODC*, pages 249—256. ACM Press, 2011.