Facility Location: Distributed Approximation

Thomas Moscibroda
Computer Engineering and
Networks Laboratory, ETH Zurich
8092 Zurich, Switzerland
moscitho@tik.ee.ethz.ch

Roger Wattenhofer
Computer Engineering and
Networks Laboratory, ETH Zurich
8092 Zurich, Switzerland
wattenhofer@tik.ee.ethz.ch

ABSTRACT

In this paper, we initiate the study of the approximability of the facility location problem in a distributed setting. In particular, we explore a trade-off between the amount of communication and the resulting approximation ratio. We give a distributed algorithm that, for every constant k, achieves an $O(\sqrt{k}(m\rho)^{1/\sqrt{k}} \log(m+n))$ approximation in $O(k)$ communication rounds where message size is bounded to $O(\log n)$ bits. The number of facilities and clients are m and n, respectively, and ρ is a coefficient that depends on the cost values of the instance. Our technique is based on a distributed primal-dual approach for approximating a linear program, that does not form a covering or packing program.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*;
G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*;
G.2.2 [**Discrete Mathematics**]: Graph Theory—*network problems*

General Terms

Algorithms, Theory

Keywords

facility location, distributed approximation, linear programming, primal-dual algorithms

1. INTRODUCTION

During the last few years, the study of *distributed approximation* has attracted a lot of attention and has resulted in several fundamental results that shed new light into the possibilities and limitations of distributed computing [6]. The interest in *distributed approximation* appears natural considering that it lies on the boundary between two well-established and important areas in computer science: *distributed computing* and *approximability*. In the same way as the theory of approximation has lead to an understanding of principles in complexity theory, the study of distributed approximation has the potential of providing a deeper understanding of the underlying distributed models [19, 5, 16].

In this paper, we investigate the distributed approximability of one of the most studied problems in operations research and the theory of approximation, the *facility location* problem. In the facility location problem, there is a set of *clients* (a.k.a. cities or demands) and a set of possible server locations, called *facilities*. Every client must be connected to a facility that serves the client's demand. Opening a facility i causes *opening costs* f_i and connecting a client j to an opened facility i incurs *connection costs* c_{ij}. The goal is to open a subset of the facilities and connect each client to an opened facility in such a way that minimizes the sum of connection costs and opening costs.

The facility location problem captures a large variety of important application scenarios. Traditionally, it has been used to model the problem of finding the best geographic location for the construction of industrial facilities or warehouses. While this classic application can be satisfactorily solved by a centralized algorithm, there are numerous applications that explicitly demand for *distributed algorithms*. Consider for instance the problem of dynamically setting up servers or placing caches in the Internet for a certain application. Setting up a server at a host in the Internet incurs overhead, traffic, and maintenance costs at that particular host. On the other hand, every client demands to access its data from a server that is as close as possible in order to minimize its delay. The resulting trade-off between the number of servers to be installed and the propagation delay maps precisely to the facility location problem.

Another example comes from the world of battery powered wireless ad hoc and sensor networks that typically feature tight energy constraints. In this context, structuring the network into energy-efficient clusters plays a key role for prolonging the networks lifetime, e.g., [10, 15]. Only selected cluster-leaders must remain active, while all other nodes can go into an energy-efficient *sleep mode* thus saving valuable battery power. Again, the trade-off to be optimized follows along the same lines. It is desirable to have as few cluster-leaders as possible since this in turn allows more nodes to go into sleep mode. However, having few cluster-heads naturally increases the distance between clusterheads and their associated nodes. This forces nodes to set their transmission power to higher values in order to reach their clusterhead.

PODC'05, July 17–20, 2005, Las Vegas, Nevada, USA.

In both of the above examples, centralized algorithms based on maintaining a global view of the network cannot be applied because no node in the network has total knowledge. In large-scale distributed Internet applications or wireless sensor networks, collecting and maintaining a global view of the network would cause a horrendous overhead in terms of both time and message complexity. Hence, nodes must come up with a solution in a distributed way.

The study of distributed approximation explores the trade-off between the amount of communication between nodes in the network, and the quality of the global solution they achieve. Specifically, we want to be able to come up with an algorithm that gives a non-trivial approximation ratio for any (even constant!) number of communication rounds k. From a theoretical point of view, having such a complete characterization of the above mentioned trade–off yields a deeper understanding of the nature of the problem. Moreover, algorithms having a constant running time independent of the size of the problem instance are often the only acceptable choice in distributed settings. This is the case if either individual nodes do not know about the size of the entire solution or, as in the case of mobile wireless networks, low bandwidth and high dynamics preclude algorithms with high running time.

Our algorithm is based on approximating the LP relaxation of the facility location problem in a distributed way. In fact, it follows a kind of *distributed primal-dual approach*. Starting with a sub-optimal but feasible primal solution and an infeasible dual solution, the nodes successively increase the primal optimality and reduce the dual infeasibility. In a second step, the obtained fractional solution to the LP is then rounded in a distributed way to a feasible integer solution to the original facility location problem.

Initiated by Papadimitriou and Yannakakis in [23], the distributed approximation of linear programs has attracted the interest of researchers for some time, e.g., [3, 17, 16]. All of these papers consider the special class of covering or packing linear programs. The distributed complexity of more general linear programs has remained a long-standing open problem since [23]. Moreover, of the above works, only [17] gives a complete characterization of the communication-quality of the solution trade-off.

In contrast, in this paper, we take a step towards understanding a more general case of LPs by presenting a characterization for the facility location problem, which does *not* form a covering or packing pair of LPs. To the best of our knowledge, our paper is the first to provide a result on the distributed approximability of *non-positive linear programs* in a constant number of communication rounds. Specifically, we consider the classic bounded message size model in which every message is restricted to $O(\log n)$ bits and the ID space is assumed to be polynomial in n, e.g. [24, 25, 5, 20]. We present an algorithm that, for arbitrary positive integers k, in $O(k)$ communication rounds, obtains an approximation ratio of $O(\sqrt{k}(m\rho)^{1/\sqrt{k}}\log(m+n))$, where m and n are the number of facilities and clients, respectively, and ρ is a parameter that depends on the coefficients of the given facility location instance. Furthermore, at the cost of a slightly worse approximation, it is possible to get rid of the dependency on ρ. This result shows that even in a constant number of communication rounds, the facility location problem can be approximated with a non-trivial approximation ratio.

The remainder of the paper is organized as follows. Section 2 gives an overview over related work and discusses the reasons why existing centralized solutions cannot be easily adapted to serve our needs. In Section 3, we introduce our model of computation and formally define the facility location problem and its linear program relaxation. The distributed approximation algorithm for the LP is presented and analyzed in subsequent Sections 4 and 5. We give a procedure for rounding the fractional LP solution to a feasible integer solution in Section 6. Section 7 generalizes the problem and shows how the dependency on ρ can be avoided. Subsequent Section 8 discusses an extension to the problem before Section 9 concludes the paper.

2. RELATED WORK

Its wide applicability and appealing simplicity have rendered *uncapacitated facility location* one of the most well-studied optimization problems in the literature [2, 11, 4]. It has not only occupied a central place in operations research, but has recently attracted a lot of attention from the perspective of approximation theory [27, 14, 13, 12, 9].

For the general *non-metric case*, Hochbaum [11] showed that the greedy algorithm is an $O(\log n)$ approximation. Set cover being a special case of facility location, this is asymptotically optimal unless $NP \subseteq DTIME(n^{O(\log\log n)})$ [22, 7]. The *filtering* technique introduced by Lin and Vitter [18] yields another $O(\log n)$ approximation algorithm. In the metric facility location problem, it is assumed that the connection costs obey the triangle inequality. In that case, the problem remains NP-hard, but constant approximations become possible. The first algorithm achieving a constant approximation ratio was given in [27]. Ever since, a flurry of research activity has lead to various improvements. Also, numerous variants of facility location have been studied, e.g. [28, 9].

Considering the vast literature on the facility location problem, surprisingly little is known about the important *distributed* case. In a seminal paper, Jain and Vazirani [13] claim that their primal-dual algorithm for the metric case of the facility location problem was also suitable in a distributed setting. However, this is only the case if either message-size is unbounded[1], or the algorithm's time-complexity depends on the size of the problem instance. That is, their primal-dual algorithm cannot be applied if the number of communication rounds is restricted to an arbitrary constant.

It is interesting to relate our work to the wider context of distributed approximation of linear programs. Starting from [23], there have been a number of efficient distributed algorithms for approximating *covering and packing LPs* [21, 3, 26, 17] and in [16] a lower bound on the distributed time-complexity of covering LPs is given. This multiplicity of results on covering and packing problems is in sharp contrast to the case of more general LPs, i.e., *non-positive linear programs*. For problems such as facility location, the achievable time-approximation trade-off has been an open question.

[1]In a complete bipartite graph, 2 communication rounds suffice to inform every client and every facility about the entire problem instance if the message size is unbounded. The problem can then be solved locally using the standard centralized greedy algorithm [11].

3. MODEL

In the formal model, the facility location instance is represented by a complete bipartite graph $G = (C \cup F, E)$. C and F denote the set of clients and facilities, respectively. Let $n = |C|$ and $m = |F|$ denote the number of clients and facilities, respectively. The non-negative opening costs of facility $i \in F$ are denoted by f_i. The connection costs between facility $i \in F$ and client $j \in C$ are denoted by c_{ij}. Notice that we do not assume the connection costs c_{ij} to form a metric. In particular, c_{ij} may be infinitely large. For ease of presentation, we make the assumption that $c_{ij} \geq 1$ and $f_i \geq 1$ for all $i \in F, j \in C$ in Sections 4 and 5. We show how to deal with costs less than 1 in Section 7. Each client and facility has a distinct ID of size $O(\log n)$ bits.

We consider a classic message passing model (e.g. [24, 25]) in which a node can send a message of size $O(\log n)$ bits to each neighbor in every communication step. To simplify the presentation of the algorithm, we assume a *synchronous* communication model. In this model, the computation is assumed to advance in global *rounds*. In each round, each client can send a message to each facility, and each facility can send a message to each client. We emphasize that the algorithm works for the *asynchronous model* by applying an appropriate synchronizer [1]. The time-complexity of an algorithm is the number of communication rounds needed by the algorithm.

Notice that even though we consider a complete bipartite graph, solving the facility location problem is not trivial. Due to the restriction in message size, any straightforward centralized approach fails to solve the problem. The most basic idea is to elect a leader v_ℓ among the facilities and send the entire information about the specific problem instance to v_ℓ. Using the standard greedy algorithm with an approximation ratio of $\log |F|$, v_ℓ could solve the instance and inform every facility about its decision. Unfortunately, such a simple centralized solution fails because shipping the entire information on the problem instance (i.e., nm connection costs and m facility costs) inherently requires time linear in the number of facilities, i.e., $O(m)$. Hence, using more sophisticated and *distributed* techniques is inevitable for designing fast algorithms.

The facility location can be described as an integer linear program (ILP), where y_i indicates if facility i is opened, and x_{ij} indicates if client j is connected to the open facility i [2]. By relaxing the integer constraints of the variables, we obtain the following integer linear program (LP).

$$
\begin{array}{llll}
\min & \sum_{i \in F} f_i y_i + \sum_{i \in F} \sum_{j \in C} c_{ij} x_{ij} & & \\
& \sum_{i \in F} x_{ij} \geq 1 & & , \forall j \in C \\
& y_i - x_{ij} \geq 0 & & , \forall j \in C, i \in F \\
& x_{ij}, y_i \in \{0, 1\} & & , \forall j \in C, i \in F
\end{array}
$$

The first constraint ensures that each client $j \in C$ is assigned to some facility $i \in F$. The second constraint guarantees that a client j can be assigned only to an open facility i. As usual, we obtain the LP-relaxation by relaxing the integer constraints to $y_i \geq 0$ and $x_{ij} \geq 0$. The relaxed dual program (DLP) is:

$$
\begin{array}{llll}
\max & \sum_{j \in C} \alpha_j & & \\
& \alpha_j - \beta_{ij} \leq c_{ij} & & , \forall j \in C, i \in F \\
& \sum_{j \in C} \beta_{ij} \leq f_i & & , \forall i \in F \\
& \alpha_j, \beta_{ij} \geq 0 & & , \forall j \in C, i \in F
\end{array}
$$

Notice that this primal and dual pair of LPs have negative coefficients and do not form a covering-packing pair.

4. ALGORITHM

The algorithm consists of two parts. First, the facilities and clients compute an approximate solution to the fractional linear program (LP), essentially following a distributed primal-dual approach. Throughout the algorithm, clients and facilities keep track of the value of their primal variables. Specifically, after $O(k)$ communication rounds, the first phase of the algorithm ends with every facility having opening value y_i and every client having connection values x_{ij} to facilities. Note that while these values constitute a feasible solution to (LP), they may be fractional and can therefore not be used as a solution to the original facility location problem. Hence, a distributed randomized rounding method described in Section 6 then rounds these fractional values to values in $\{0, 1\}$, increasing the approximation ratio from the fractional solution only by a logarithmic factor. The technique of designing distributed approximation algorithms by first computing a fractional solution and rounding them in a second phase has been inspired by [17]. Our algorithm combines these techniques with ideas from the centralized primal-dual approach in [13].

At the heart of our algorithm is the distributed primal-dual technique which deterministically approximates (LP) within a constant number of communication rounds. Each facility and client executes Algorithm 1 and 2, respectively. The algorithms consist of two nested loops which are both executed $h = \lceil \sqrt{k} \rceil$ times. The number of communication rounds in each iteration of the inner loop is constant, yielding the claimed constant time-complexity of $O(k)$. Initially, all primal and dual variables y_i, x_{ij}, α_j, and β_{ij} are set to zero. Hence, the initial primal solution is infeasible, and the dual solution is feasible, yet far from optimal. During the course of the algorithm, both the primal and dual variables are gradually increased, thereby decreasing the primal infeasibility, and increasing dual optimality. At the end of the h^{th} iteration of the outer loop, the primal variables y_i and x_{ij} form a feasible solution to (LP).

A client j is called *uncovered* if it is not yet (fractionally) connected to one facility, i.e., $\sum_{i \in F} x_{ij} < 1$. At any moment throughout the algorithm, the set of uncovered clients is called the *uncovered set* $\mathcal{A}$, initially $\mathcal{A} = C$. Whenever a client j becomes covered, it sends a message $\mathcal{M}_j$ to the facilities. That way, the facilities always have a consistent view of the current $\mathcal{A}$ (Lines 8 and 9 of the algorithm).

A *star* consists of one facility $i \in F$ and several uncovered clients $j \in \mathcal{A}$. The cost efficiency of a star B is the sum of the connection costs of the clients to facility i plus the facility cost f_i divided by the number of clients $|B|$. The *cost efficiency* $c(i)$ of a facility i is defined as the minimum star spanned from i, i.e.,

$$c(i) := \min_{B \in 2^{\mathcal{A}}} \frac{f_i + \sum_{j \in B} c_{ij}}{|B|}. \quad (1)$$

The basic idea of the outer loop (s-loop) is to increase the y_i value of facilities with comparatively good cost efficiency $c(i)$[2]. More precisely, we call a facility *active* in a given iteration if its cost efficiency is at most $c(i) \le \rho^{s/h}$. Only active facilities, that is, only facilities with good cost efficiency will execute the code between lines 11 and 22. Particularly, only active facilities will increase their y_i value during an iteration. The idea of increasing the y_i value of facilities with good cost efficiency is inspired by the centralized greedy algorithm [11] that iteratively picks the facility with the best cost efficiency. In order to come up with fast (and particularly constant time) algorithms in a distributed setting, this "greedy step" has to be parallelized. However, the greedy step's parallelization must be carefully implemented in order to avoid opening too many facilities at once, thus overly deteriorating the algorithm's performance.

We call a client j *tight* to an active facility i in iteration s of the outer loop if $c_{ij} \le \rho^{s/h}$. That is, the *tight set* T_i in line 12 consists of all clients that are connected to i by a connection of cost at most $\rho^{s/h}$. The significance of the tight set is that the increase of y_i in a given iteration results in an identical increase of x_{ij} of all clients j being in the tight set T_i. Since a client j may concurrently be in the tight set of several facilities, the increase of the different y_i must be handled with care. This is the role of the inner loop (t-loop), during which the y_i are gradually increased (line 19) as long as the facility remains active. Finally, note that 2 communication rounds suffice for every facility to compute the value $\rho = \max_{j \in C} \min_{i \in F} (c_{ij} + f_i)$ in Line 3.

Algorithm 1 Facility i

1: $h := \lceil \sqrt{k} \rceil$;
2: **receive** c_{ij} from all $j \in C$;
3: $\rho := \max_{j \in C} \min_{i \in F} (c_{ij} + f_i)$;
4: $y_i := 0$; $\mathcal{A} := C$;
5: **for** $s := 1$ to h by 1 **do**
6: $\quad \pi_i^s := 0$;
7: $\quad$ **for** $t := h - 1$ to 0 by -1 **do**
8: $\quad\quad$ **receive** $\mathcal{M}_j$ from all $j \in C$;
9: $\quad\quad \mathcal{A} := \mathcal{A} \setminus \{j \in C \mid \mathcal{M}_j = 1\}$
10: $\quad\quad c(i) := \min_{B \in 2^{\mathcal{A}} \setminus \{\}} \frac{f_i + \sum_{j \in B} c_{ij}}{|B|}$;
11: $\quad\quad$ **if** $c(i) \le \rho^{s/h}$ **then**
12: $\quad\quad\quad T_i := \{j \in \mathcal{A} \mid c_{ij} \le \rho^{s/h}\}$
13: $\quad\quad\quad \Gamma_i := (f_i + \sum_{j \in T_i} c_{ij})/|T_i|$
14: $\quad\quad\quad$ **if** $t = h - 1$ **then**
15: $\quad\quad\quad\quad T_i^s := T_i$;
16: $\quad\quad\quad\quad \Gamma_i^s := \Gamma_i$;
17: $\quad\quad\quad\quad \pi_i^s := 1$;
18: $\quad\quad\quad$ **end if**
19: $\quad\quad\quad \Delta y_i := \max\{y_i, m^{-t/h}\} - y_i$
20: $\quad\quad\quad y_i := y_i + \Delta y_i$;
21: $\quad\quad\quad$ **send** $(\Delta y_i, \Gamma_i)$ to all $j \in T_i$
22: $\quad\quad$ **end if**
23: $\quad$ **end for**
24: $\quad$ **forall** $i \in T_i^s$ **do**
25: $\quad\quad \Delta\beta_{ij} := \begin{cases} 0 & , \rho^{s/h} < c_{ij} \\ \rho^{s/h} - c_{ij} & , \rho^{s/h} \ge c_{ij} \wedge \pi_i^s = 0 \\ \Gamma_i^s - c_{ij} & , \rho^{s/h} \ge c_{ij} \wedge \pi_i^s = 1 \end{cases}$
26: $\quad\quad \beta_{ij} := \beta_{ij} + \Delta\beta_{ij}$
27: $\quad$ **end for**
28: **end for**

Algorithm 2 Client j

1: $h := \lceil \sqrt{k} \rceil$;
2: **send** c_{ij} to all $i \in F$;
3: $\alpha_j := 0$;
4: $\forall i \in F : x_{ij} := 0$;
5: **for** $s := 1$ to h by 1 **do**
6: $\quad$ **for** $t := h - 1$ to 0 by -1 **do**
7: $\quad\quad \mathcal{M}_j := 0$;
8: $\quad\quad$ **if** $\sum_{i \in F} x_{ij} \ge 1$ **then** $\mathcal{M}_j := 1$;
9: $\quad\quad$ **send** $\mathcal{M}_j$ to all $i \in F$;
10: $\quad\quad$ **receive** $(\Delta y_i, \Gamma_i)$ from all $i \in T_j$;
11: $\quad\quad$ **forall** $i \in F$ **do** $\Delta x_{ij} := \Delta y_i$;
12: $\quad\quad x_{ij} := x_{ij} + \Delta x_{ij}$;
13: $\quad\quad \Delta\alpha_j := \sum_{i: j \in T_i} \Delta y_i \Gamma_i$;
14: $\quad\quad \alpha_j := \alpha_j + \Delta\alpha_j$;
15: $\quad$ **end for**
16: **end for**

5. ANALYSIS

In a sense, our algorithm's analysis is based on the method of *dual fitting* [12] applied in a distributed setting. The basic idea of this method applied to FL can be described as follows: Using the linear program relaxation (LP) for facility location and its dual (DLP), we interpret our combinatorial algorithm as an algorithm that iteratively makes primal and dual updates in a distributed fashion. Unfortunately, these updates do generally not lead to a feasible dual solution. However, the idea is to show that the objective function of the primal fractional solution computed by the algorithm is bounded by that of the dual. That is, the primal solution is fully paid for by the dual. By the basic laws of LP duality, it then remains to divide all dual values by a suitably large factor α that renders the dual variables feasible. The shrunk dual objective function is then a lower bound on OPT, and α is the algorithm's approximation guarantee. That is, instead of relaxing complementary slackness conditions as done in other primal-dual algorithms (e.g., [8, 29]), we relax the *feasibility* of the dual itself.

For notational clarity, we denote the increase of Δy_i in a certain iteration of the s and t-loop by $\Delta y_i(s,t)$ throughout this section. $\Delta x_{ij}(s,t)$, $\Delta\alpha_i(s,t)$, and $\Delta\beta_{ij}(s)$ are defined analogously.

We begin the analysis with the observation that the resulting primal solution is feasible.

LEMMA 5.1. *Algorithms 1 and 2 produces a feasible primal solution for (LP).*

PROOF. The feasibility of the second LP condition, $y_i - x_{ij} \ge 0$, $\forall j \in C, i \in F$, directly follows from the definition of the algorithm. Specifically, in Line 11 of Algorithm 2, the

[2]In spite of there being exponentially many sets $B \in 2^{\mathcal{A}}$, the facility can compute its cost efficiency $c(i)$ in polynomial time in Line 10 of Algorithm 1 by considering the clients ordered according to their connection costs c_{ij}.

increase of a connection variable, Δx_{ij} never exceeds the increase of the corresponding y_i.

As for the first LP condition, assume for contradiction that j is a client which is still uncovered at the end of the algorithm, i.e., $\sum_{i \in F} x_{ij} < 1$ and $j \in \mathcal{A}$. Now, consider the very last iteration of the inner loop ($s = h$, $t = 0$). By the definition of ρ, there exists at least one facility i with cost efficiency $c(i) \leq \rho$ covering client j. Because $s = h$, facility i will become *active* and increase its y_i value to $m^{-t/h} = 1$ in Lines 19 and 20. Subsequently, j will set $x_{ij} := 1$ which contradicts the assumption that $j \in \mathcal{A}$ at the end of the algorithm. □

If a facility is active in a certain iteration, its cost-efficiency $c(i)$ is, by definition, at most $\rho^{s/h}$. The tight set T_i does not necessarily contain the same clients which constituted the optimal cost-efficiency. Therefore, the cost efficiency of T_i may be larger than $c(i)$, i.e., $\Gamma_i \geq c(i)$. The next lemma shows that the cost-efficiency of the tight set T_i, Γ_i, is at most $\rho^{s/h}$.

Lemma 5.2. *In every iteration of the t-loop, if $c(i) \leq \rho^{s/h}$ for a facility i, then $\Gamma_i \leq \rho^{s/h}$.*

Proof. Consider the set B that constituted $c(i)$. First, observe that if $c(i) \leq \rho^{s/h}$, and because B minimizes $c(i)$, no client $j \in B$ can have connection cost $c_{ij} > \rho^{s/h}$. Let $Q := T_i \setminus B$ be the set of clients $j \notin B$ with $c_{ij} \leq \rho^{s/h}$. Γ_i is upper bounded by

$$\begin{aligned} \Gamma_i &= \frac{f_i + \sum_{j \in T_i} c_{ij}}{|T_i|} \\ &= \frac{f_i + \sum_{j \in B} c_{ij} + \sum_{j \in Q} c_{ij}}{|T_i|} \\ &\underset{c(i) \leq \rho^{s/h}}{\leq} \frac{|B|\rho^{s/h} + |Q|\rho^{s/h}}{|T_i|} = \rho^{s/h}. \end{aligned}$$

□

Bounding the primal objective function by the dual objective function is key to applying the method of dual fitting. The next lemma provides such a bound by showing that throughout the execution of the algorithm, the values of the primal and dual objective functions are equal.

Lemma 5.3. *At the end of each t-loop, it holds that*

$$\sum_{j \in C} \alpha_j = \sum_{j \in C, i \in F} c_{ij} x_{ij} + \sum_{i \in F} f_i y_i. \tag{2}$$

Proof. We prove the claim by induction over the iterations of the t-loop. At the beginning of the algorithms, both sides of the equation are 0. Assume that the claim is true before starting a new iteration s'. If facility i increases its y_i during s', all tight clients $j \in T_i$ increase their corresponding x_{ij} as well. Hence, the right hand side of (2) increases by

$$\Delta RHS = \sum_{i \in F} \Delta y_i f_i + \sum_{i \in F} \sum_{j \in T_i} \Delta y_i c_{ij}.$$

As for the left hand side of (2), the value $\sum_{j \in C} \alpha_j$ increases by

$$\begin{aligned} \Delta LHS &= \sum_{j \in C} \sum_{i: j \in T_i} \Delta y_i \Gamma_i = \sum_{i \in F} \sum_{j \in T_i} \Delta y_i \Gamma_i \\ &= \sum_{i \in F} \Delta y_i \left(\sum_{j \in T_i} (f_i + \sum_{j \in T_i} c_{ij}) / |T_i| \right) \\ &= \sum_{i \in F} \Delta y_i \cdot \left(f_i + \sum_{j \in T_i} c_{ij} \right) \\ &= \Delta RHS. \end{aligned}$$

□

In the next lemma, we characterize the steady increase of a facility i's cost efficiency during the course of the algorithm.

Lemma 5.4. *At the beginning of each iteration of the s-loop, it holds for all facilities $i \in F$ that $c(i) \geq 1$ for $s = 1$ and $c(i) > \rho^{s-1/h}$ for $s > 1$.*

Proof. The case $s = 1$ follows from the assumption that $c_{ij} \geq 1$ and $f_i \geq 1$ (cf. Section 7). Consider iteration $s > 1$ and let $s' = s - 1$. In the last t-loop iteration of the s'^{th} iteration, all facilities i with $c(i) \leq \rho^{s'/h}$ set y_i to 1. Consequently, all $j \in T_i$ become covered. It follows that for such a facility i, $\forall j \in \mathcal{A}: \; c_{ij} > \rho^{s'/h}$. The claim now follows from

$$\begin{aligned} c(i) &= \min_{B \in 2^{\mathcal{A}} \setminus \{\}} \frac{f_i + \sum_{j \in B} c_{ij}}{|B|} \\ &> \frac{f_i}{|B|} + \frac{|B|\rho^{s-1/h}}{|B|} > \rho^{s-1/h}. \end{aligned}$$

□

A client may be tight with several facilities. If all these facilities increased their y_i values, the dual value α_j of j may increase too much. Consider an iteration of the s-loop. During the early iterations of the t-loop, the increase in Δy_i of active facilities is small, because t is close to h. Intuitively, it is acceptable if a client is tight to many active facilities in these early iterations. In other words, the higher the increases Δy_i of active facilities, the fewer active facilities a client is allowed to be tight to. The following lemma establishes precisely this relationship.

Lemma 5.5. *Let $A_j := \{i \mid j \in T_i\}$ be the active set for an uncovered client j. At the beginning of each iteration of the t-loop,*

$$|A_j| \leq m^{t+1/h}. \tag{3}$$

Proof. From the previous iteration of the loop, we know that for each active facility $i \in A_j$, it holds that $y_i \geq m^{-(t+1)/h}$. Now, assume for contradiction that $|A_j| > m^{t+1/h}$ for some $j \in C$. If so, then

$$\sum_{i \in A_j} y_i \geq |A_j| \cdot m^{-(t+1)/h} > 1$$

and consequently $\sum_{i \in F} x_{ij} > 1$. This contradicts the assumption that client j is uncovered. Hence, the claim follows. □

In the next lemma, we bound the amount of $\Delta\alpha_j$ that each client can receive in one iteration of the s-loop. For that purpose, let $\Delta\alpha_j(s) := \sum_{t=0}^{h-1} \Delta\alpha_j(s,t)$ be the increase of α_j during the s^{th} iteration of the outer loop. Let $T_i(s,t)$ be the set of clients that are tight to i in the iterations s and t. Further, we define

$$\sigma_j(s) := \sum_{t=1}^{h} \sum_{i \in A_j(s,t)} \Delta y_i.$$

Intuitively, $\sigma_j(s)$ is the increase of the y_i value at facilities to which client j has been tight during the course of the s^{th} iteration of the outer loop. The following lemma relates $\alpha_j(s)$ and $\sigma_j(s)$.

Lemma 5.6. *The sum of the $\Delta\alpha_j$ values collected in iteration s at a node j is upper bounded by*

$$\Delta\alpha_j(s) \;\leq\; \sigma_j(s) \cdot \rho^{s/h}.$$

Proof. Applying Lemma 5.2 and by the definition of $\Delta\alpha_j$, we have

$$\begin{aligned}
\Delta\alpha_j(s) &\leq \sum_{t=0}^{h-1} \Delta\alpha_j(s,t) \\
&\leq \sum_{t=0}^{h-1} \sum_{i \in A_j(s,t)} \Delta y_i(s,t) \Gamma_i(s,t) \\
&\underset{\text{Lemma 5.2}}{\leq} \sum_{t=0}^{h-1} \sum_{i \in A_j(s,t)} \Delta y_i(s,t) \rho^{s/h} \\
&= \sigma_j(s) \rho^{s/h}.
\end{aligned}$$

□

Next, we want to find bounds for $\sigma_j(s)$. Assume that client j becomes covered during iteration s_j^* of the outer loop. Notice that for every client j, there is exactly one iteration s_j^*. Once covered, j will not be in $\mathcal{A}$ and therefore, not in any T_i. Consequently, $\sigma_j(s') = 0$ for all $s' > s_j^*$. The other two cases, $s' < s_j^*$ and $s' = s_j^*$ are subject of the following lemma.

Lemma 5.7. *For all iterations of the s-loop, it holds that*

$$\begin{aligned}
\sigma_j(s) &\leq 1 && \forall s \neq s_j^* \\
\sigma_j(s) &\leq m^{1/h} && s = s_j^*
\end{aligned}$$

Proof. The first case, $s' < s_j^*$, follows from the definition of $\sigma_j(s)$. If

$$\sigma_j(s') \;=\; \sum_{t=1}^{h} \sum_{i \in A_j(s,t)} \Delta y_i \;\geq\; 1,$$

then j would have become covered in iteration s' and hence, $s' = s_j^*$.

It remains to analyze the iteration during which j becomes covered, i.e., $s' = s_j^*$. Consider the iterations of the inner loop during iteration s_j^* of the outer loop. Let t^* denote the iteration during which j becomes covered. Clearly, for all $t' > t^*$, it holds that $\sum_{i: j \in T_i(t')} \Delta y_i(t') = 0$ because j is already covered. Hence, we only need to analyze the first t^* iterations of the inner loop. Summing up all increases, we get

$$\begin{aligned}
\sigma_j(s') &= \sum_{t=t^*+1}^{h-1} \sum_{i \in A_j(t)} \Delta y_i(t) + \sum_{i \in A_j(t^*)} \Delta y_i(t^*) \\
&\leq 1 + \sum_{i \in A_j(t^*)} \Delta y_i(t^*) \\
&\leq 1 + \left(m^{-t/h} - m^{-(t+1)/h}\right) \cdot |A_j(t^*)| \\
&\leq 1 + \left(m^{-t/h} - m^{-(t+1)/h}\right) \cdot m^{t+1/h} \\
&\leq 1 + \left(m^{1/h} - 1\right) = m^{1/h}.
\end{aligned}$$

The first inequality follows from the fact that by definition of t^*, client j is not covered after the iterations $h-1, \ldots, t^*+1$. The third inequality follows from Lemma 5.5. □

For our dual solution to be feasible, the linear program condition imposes that $\sum_{j \in C} \beta_{ij} \leq f_i$ holds for all facilities $i \in F$. Unfortunately, the dual solution produced by our algorithm does not exhibit this feasibility property. However, we can at least show that the degree of infeasibility is bounded. Specifically, it holds that if we only consider the sum of the increases of the β_{ij} in *a single iteration* of the s-loop, it fulfils the desired property.

Lemma 5.8. *For all $i \in F$ and all iterations s of the outer loop, it holds that*

$$\sum_{j \in C} \Delta\beta_{ij}(s) \;\leq\; f_i.$$

Proof. We distinguish two cases, depending on whether $\pi_i(s)$ equals 0 or 1. In the first case, $\pi_i(s) = 0$, the facility i's cost efficiency was insufficient to increase its y_i value during the s^{th} iteration. We therefore have

$$\begin{aligned}
\sum_{j \in C} \Delta\beta_{ij}(s) &= \sum_{j \in T_i} (\rho^{s/h} - c_{ij}) \\
&= \rho^{s/h} |T_i| - \sum_{j \in T_i} c_{ij}
\end{aligned}$$

Assume for contradiction that $\sum_{j \in C} \Delta\beta_{ij}(s) > f_i$ for some facility i and iteration s. It follows that

$$\rho^{s/h} \;>\; \frac{f_i + \sum_{j \in T_i} c_{ij}}{|T_i|} \;\geq\; c(i),$$

which in turn implies $\pi_i(s) = 1$ for $B = T_i$. This establishes the contradiction.

As for the second case, $\pi_i(s) = 1$, we have

$$\begin{aligned}
\sum_{j \in C} \Delta\beta_{ij}(s) &= \sum_{j \in T_i} (\Gamma_i^s - c_{ij}) \\
&= |T_i| \cdot \frac{f_i + \sum_{j \in T_i} c_{ij}}{|T_i|} - \sum_{j \in T_i} c_{ij} \;=\; f_i.
\end{aligned}$$

Therefore, the lemma holds in both cases. □

Having bounded the degree of infeasibility of the first dual constraint, it now remains to do the same for the second one, $\alpha_j - \beta_{ji} \leq c_{ij}$, for all $j \in C$ and $i \in F$. Again, we give weaker bounds that do not hold for the entire execution of the algorithm, but merely for a single iteration of the outer loop.

LEMMA 5.9. *Let $\Delta\alpha_j(s)$ be the sum of the $\Delta\alpha_j(t)$ over all iterations of the t-loop in the s^{th} iteration of the s-loop. For all $j \in C$, $i \in F$, and all iterations s, it holds that*

$$\frac{\Delta\alpha_j(s)}{(m\rho)^{1/h}} - \Delta\beta_{ij}(s) \quad \leq \quad c_{ij}$$

PROOF. We distinguish three cases, depending on how much increase of β_{ij} was assigned to the connection between i and j in line 25 of the facility algorithm. Regardless of the specific case, the value $\Delta\alpha_j(s)$ is bounded by $\Delta\alpha_j(s) \leq \rho^{s/h} m^{1/h}$ by Lemmas 5.6 and 5.7.

1) In the case $\rho^{s/h} \leq c_{ij}$, the algorithm sets $\Delta\beta_{ij}(s)$ to 0. Therefore

$$\frac{\Delta\alpha_j(s)}{(m\rho)^{1/h}} - \Delta\beta_{ij}(s) \ \leq \ \rho^{(s-1)/h} \ \leq \ c_{ij}.$$

2) In the second case, the client j is tight to i, i.e., $\rho^{s/h} > c_{ij}$, but $\pi_i(s) = 0$. Plugging in the corresponding value for $\Delta\beta_{ij}(s)$, we get

$$\begin{aligned} \frac{\Delta\alpha_j(s)}{(m\rho)^{1/h}} - \left(\rho^{s/h} - c_{ij}\right) \quad &\leq \quad \rho^{(s-1)/h} - \rho^{s/h} + c_{ij} \\ &\leq \quad c_{ij}. \end{aligned}$$

3) Finally, consider the last case, $\rho^{s/h} > c_{ij}$ and $\pi_i(s) = 1$. Substituting $\Delta\beta_{ij}(s)$ by $\Gamma_i^s - c_{ij}$ yields

$$\Delta\beta_{ij}(s) + c_{ij} \ = \ \Gamma_i^s - c_{ij} + c_{ij} \ = \ \Gamma_i^s \ \geq \ c(i).$$

For $s > 1$, we know by Lemma 5.4, that $c(i) \geq \rho^{(s-1)/h}$ at the beginning of iteration s, hence

$$\Delta\beta_{ij}(s) + c_{ij} \ \geq \ \rho^{(s-1)/h}.$$

Using the above inequality, we obtain

$$\frac{\Delta\alpha_j(s)}{(m\rho)^{1/h}} \quad \leq \quad \rho^{(s-1)/h} \ \leq \ \Delta\beta_{ij}(s) + c_{ij}.$$

Subtracting $\Delta\beta_{ij}(s)$ concludes the proof for the case $s > 1$. The case $s = 1$ follows similarly. By Lemma 5.4, we can lower bound $c(i) \geq 1$ and therefore $\Delta\beta_{ij}(s) + c_{ij} \geq 1$. The claim now follows from

$$\frac{\Delta\alpha_j(s)}{(m\rho)^{1/h}} \ \leq \ \rho^{(s-1)/h} \ = \ 1 \ \leq \ \Delta\beta_{ij}(s) + c_{ij}.$$

□

Having bounded the degree of dual infeasibility in the two previous lemmas, we can now establish the approximation ratio of the algorithm using the laws of LP duality. Specifically, we prove that the dual feasibility is violated only by a factor $O(h(m\rho)^{1/h})$ and hence, when dividing α_j and β_{ij} by suitably large values, we obtain a feasible solution $\hat{\alpha}_j$ and $\hat{\beta}_{ij}$.

THEOREM 5.10. *For an arbitrary integer $k > 0$, the algorithm computes a $O(\sqrt{k}(m\rho)^{1/\sqrt{k}})$ approximation to the fractional facility location problem in $O(k)$ communication rounds.*

PROOF. The runtime follows directly from the definition of the algorithm. For the analysis of the approximation ratio, we define $\hat{\alpha}_j$ and $\hat{\beta}_{ij}$ as

$$\hat{\alpha}_j := \frac{\alpha_j}{h(m\rho)^{1/h}} \quad \text{and} \quad \hat{\beta}_{ij} := \frac{\beta_{ij}}{h},$$

respectively. We show that the variables $\hat{\alpha}_j$ and $\hat{\beta}_{ij}$ form a feasible solution to the dual LP. The feasibility of the second dual constraint follows directly from Lemma 5.8. Particularly, it holds that $\sum_{j\in C} \beta_{ij}(s) \leq f_i$ for all iterations s and all facilities i. As a consequence, we obtain $\sum_{j\in C} \beta_{ij} \leq h \cdot f_i$ and therefore, $\sum_{j\in C} \hat{\beta}_{ij} \leq f_i$.

Next, we show the feasibility of the first constraint by bounding $\hat{\alpha}_j - \hat{\beta}_{ij}$ as

$$\begin{aligned} \hat{\alpha}_j - \hat{\beta}_{ij} \quad &= \quad \frac{\sum_{s=0}^{h-1} \alpha_j(s)}{h(m\rho)^{1/h}} - \frac{\sum_{s=0}^{h-1} \beta_{ij}(s)}{h} \\ &= \quad \frac{1}{h} \sum_{s=0}^{h-1} \left(\frac{\alpha_j(s)}{(m\rho)^{1/h}} - \beta_{ij}(s) \right). \end{aligned}$$

By Lemma 5.9, each term of the sum is bounded by c_{ij}. Therefore, we have $\hat{\alpha}_j - \hat{\beta}_{ij} \leq hc_{ij}/h \leq c_{ij}$.

Let OPT and ALG denote the optimal value and the value as computed by the algorithm, respectively. By LP duality, the sum of the $\hat{\alpha}_j$ values is a lower bound for OPT. As for ALG, recall that by Lemma 5.3, we know that the value of the primal and dual objective function is equal at the end of the algorithm. Therefore, we can bound ALG as

$$\begin{aligned} ALG \quad &= \quad \sum_{i\in F} f_i y_i + \sum_{i\in F} \sum_{j\in C} c_{ij} x_{ij} \\ &\underset{\text{Lemma 5.3}}{=} \quad \sum_{j\in C} \alpha_j \ \leq \ h(m\rho)^{1/h} \sum_{j\in C} \hat{\alpha}_j \\ &\leq \quad h(m\rho)^{1/h} \cdot OPT. \end{aligned}$$

Finally, the theorem follows from $h = \lceil\sqrt{k}\rceil$. □

6. RANDOMIZED ROUNDING

In order to come up with a solution to the integer facility location problem, we round the fractional solutions obtained in the previous section. During this process, we must neither overly increase the total opening costs, nor the total connection costs. Interestingly, this can be achieved with high probability in constant time even in a distributed setting. The idea for the randomized rounding is based on the filtering technique introduced in [18]. Applications of randomized rounding for a covering LP in a distributed context can be found in [17].

In the following, let $\hat{x}_{ij}$ and $\hat{y}_i$ be the fractional values obtained from Algorithms 2 and 1, respectively. The variables x_{ij} and y_i denote the rounded integer values. For every client $j \in C$, let $C_j^* := \sum_{i\in F} c_{ij}\hat{x}_{ij}$ be the weighted cost of j's connections. Further, let the *neighborhood* V_j of a client j be the set of all facilities that are located within a factor of $\log(n+m)$ of the weighted connection cost. Formally, for every $j \in C$, $V_j := \{i \in F \mid c_{ij} \leq \log(n+m) \cdot C_j^*\}$. The idea is to round the fractional values y_i at each facility i in such a way that with high probability, all clients have at least one opened facility in their *neighborhood*, $N_j \neq \emptyset$. Each such client then simply connects itself to the open facility with the minimum connection cost c_{ij}.

THEOREM 6.1. *Let $\hat{x}_{ij}$ and $\hat{y}_i$ for each $j \in C$ and $i \in F$ be the fractional solution with cost at most $\alpha \cdot OPT$ as derived in Algorithms 1 and 2. In two rounds of communication, Algorithms 4 and 3 produce an integer solution x_{ij}, y_i with cost at most $O(\log(m+n))\alpha OPT$ in expectation.*

Algorithm 3 Randomized Rounding - Client

INPUT: fractional solution $\hat{x}_{ij}$ from Algorithm 2
OUTPUT: integral solution x_{ij} to ILP

$C_j^* := \sum_{i\in F} c_{ij}\hat{x}_{ij}$;
$V_j := \{i \in F \mid c_{ij} \le \ln(n+m)\cdot C_j^*\}$;
receive y_i from all $i \in F$
$N_j := V_j \cap \{i \in F \mid y_i = 1\}$
if $N_j \neq \emptyset$ **then**
  $i' := \operatorname{argmin}_{i\in N_j} c_{ij}$; $x_{i'j} := 1$;
else
  $i' := \operatorname{argmin}_{i\in F}(c_{ij} + f_i)$; $x_{i'j} := 1$;
  send JOIN-MSG to facility i';
fi

Algorithm 4 Randomized Rounding - Facility

INPUT: fractional solution $\hat{y}_i$ from Algorithm 1
OUTPUT: integral solution y_i to ILP

1: $p_i := \min\{1, \hat{y}_i \cdot \ln(n+m)\}$;
2: $y_i := \begin{cases} 1 \text{ , with probability } p_i \\ 0 \text{ , with probability } 1-p_i \end{cases}$
3: **send** y_i to all clients $j \in C$;
4: **if receive** JOIN-MSG **then** $y_i := 1$;

PROOF. It follows from the definition of C_j^* and V_j that $\sum_{j\in F\setminus V_j}\hat{x}_{ij} \le 1/\log(n+m)$, for if not, C_j^* would be larger. Since the construction of Algorithms 2 and 1 guarantees the invariant $\sum_{i\in V_j}\hat{x}_{ij} = \sum_{i\in V_j}\hat{y}_i$, and $\sum_{i\in F}\hat{x}_{ij} \ge 1$ we have

$$\sum_{i\in V_j}\hat{y}_i \;=\; \sum_{i\in V_j}\hat{x}_{ij} \;\ge\; 1 - \frac{1}{\log(n+m)}. \tag{4}$$

For each client j having $N_j \neq \emptyset$, the connection costs are at most $c_{ij} \le \ln(n+m)C_j^*$ by the definition of the neighborhood V_j. It follows that these clients account for total connection costs of at most $\ln(n+m)\sum_{j\in C, i\in F} c_{ij}\hat{x}_{ij}$. A facility declares itself open in Line 2 of Algorithm 4 with probability $\min\{1, \hat{y}_i \cdot \ln(n+m)\}$. The expected opening costs of facilities opened in Line 2 are thus bounded by the value $\ln(n+m)\sum_{i\in F}\hat{y}_i f_i$.

It remains to bound the costs incurred by clients that are *not* covered, i.e. $N_j = \emptyset$, and facilities that are opened via a JOIN-MSG message. The probability q_j that a client j does *not* have an open facility in its neighborhood is at most

$$\begin{aligned}
q_j \quad &= \quad \prod_{i\in V_j}(1-p_i) \;=\; \left(\sqrt[n+m]{\prod_{i\in V_j}(1-p_i)}\right)^{n+m} \\
&\le \quad \left(\frac{\sum_{i\in V_j}(1-p_i)}{n+m}\right)^{n+m} \\
&\underset{|V_j|\le m}{\le} \quad \left(1 - \frac{\ln(n+m)\sum_{i\in V_j}\hat{y}_i}{n+m}\right)^{n+m} \\
&\underset{\text{Eq. (4)}}{\le} \quad \left(1 - \frac{\ln(n+m)}{n+m}\left(1 - \frac{1}{\ln(n+m)}\right)\right)^{n+m} \\
&= \quad \left(1 - \frac{\ln(n+m)-1}{n+m}\right)^{n+m} \\
&\le \quad e^{-\ln(n+m)-1} \;\le\; \frac{1}{e(n+m)}.
\end{aligned} \tag{5}$$

The first inequality follows from the fact that for every sequence of positive numbers, the geometric mean is smaller than or equal to the arithmetic mean of these numbers.

An uncovered client sends a JOIN-MSG message to the facility $i \in F$ that minimizes $c_{ij} + f_i$. Each of these costs is at most $\sum_{j\in C, i\in F} c_{ij}\hat{x}_{ij} + \sum_{i\in F}\hat{y}_i f_i$ because $\hat{x}$ and $\hat{y}$ would not constitute a feasible solution otherwise. Combining this with the above results, the total expected cost $\mu = E[ALG]$ is

$$\begin{aligned}
\mu \quad &\le \quad \ln(n+m)\left(\sum_{j\in C, i\in F} c_{ij}\hat{x}_{ij} + \sum_{i\in F}\hat{y}_i f_i\right) \\
&\quad + \frac{n}{e(n+m)}\left(\sum_{j\in C, i\in F} c_{ij}\hat{x}_{ij} + \sum_{i\in F}\hat{y}_i f_i\right) \\
&\le \quad (\ln(n+m) + O(1))\,\alpha OPT.
\end{aligned}$$

This concludes the proof of Theorem 6.1. □

In many distributed systems, obtaining a solution that holds in expectation may not be satisfying. Instead, we are interested in results that hold with high probability. In order to obtain such a high probability result, the above rounding procedure can be adapted as follows.

Algorithm 3 for the clients remains unchanged. The procedure executed by each facility, however, is changed such that, instead of (probabilistically) selecting a single binary variable y_i, a facility determines a series of $\log n$ independent random variables $y_i^1, \ldots, y_i^{\log n}$. Like in Algorithm 4, each $y_i^\ell, \ell = 1, \ldots, \log n$ is independently chosen as

$$y_i^\ell := \begin{cases} 1 \text{ , with probability } p_i \\ 0 \text{ , with probability } 1-p_i \end{cases}$$

for $p_i = \min\{1, \hat{y}_i \cdot \ln(n+m)\}$.

Every facility then wraps all these $\log n$ bits in a message and sends it to some *leader* node (which can be the client with the lowest ID, for instance). The leader node receives these messages $(y_i^1 \ldots y_i^{\log n})$ from all facilities $i \in F$ and computes the index t which minimizes the sum of the opening costs, formally $t = \arg\min \sum_{i\in F} y_i^s f_i$, for all $s = 1, \ldots, \log n$. The leader then sends to each facility $i \in F$ its corresponding binary variable y_i^t, which facility i subsequently sends to its cities in Line 3 of Algorithm 4. The following Theorem shows that this adapted procedure yields the desired high probability result.

THEOREM 6.2. *Let $\hat{x}_{ij}$ and $\hat{y}_i$ for each $j \in C$ and $i \in F$ be the fractional solution with cost at most $\alpha\cdot OPT$ as derived in Algorithms 1 and 2. In four rounds of communication, the adapted rounding algorithm produces an integer solution x_{ij}, y_i with cost at most $O(\log(m+n))\alpha OPT$ with probability $1 - n^{-1}$.*

PROOF. We know that $E[X_\ell] = \ln(n+m)\sum_{i\in F}\hat{y}_i f_i$, where $X_\ell = \sum_{i\in F} y_i^\ell f_i$. By Markov's inequality,

$$P\left[X_\ell \ge 2\ln(n+m)\sum_{i\in F}\hat{y}_i f_i\right] \;\le\; \frac{1}{2}$$

for each $\ell = 1, \ldots, \log n$. The probability that X_t does not exceed this threshold is therefore bounded by

$$P\left[X_t \le 2\ln(n+m)\sum_{i\in F}\hat{y}_i f_i\right] \le 1 - \left(\frac{1}{2}\right)^{\log n} = 1 - \frac{1}{n}.$$

The rest of the proof is identical to the proof of Theorem 6.1. Particularly, we know by Inequality (5) that with high probability, the solution y_i^t for all $i \in F$ constitutes a solution such that, every client $j \in C$ has an open facility in its neighborhood. Because all clients can thus connect to open facilities in their neighborhood, the total connection cost is at most $\ln(n+m)\sum_{j\in C, i\in F} c_{ij}\hat{x}_{ij}$. We conclude the proof by observing that the message size remains in $O(\log n)$. □

Clearly, this high probability result comes at the cost of a "centralized" program execution, i.e., all information is compiled and processed at one single node. Depending on the specific application scenario, the decentralized approach of Algorithms 3 and 4 may be preferable for various reasons (including fault-tolerance) even at the cost of worse performance guarantees.

7. ARBITRARY COEFFICIENTS

The algorithm and analysis of Sections 4 and 5 is based on the assumption that $c_{ij} \geq 1$ and $f_i \geq 1$ for all $j \in C, i \in F$. In this section, we show how to handle the general case in which connection and opening costs can be arbitrary non-negative values. Furthermore, this technique can be used to get rid of the dependency on ρ in the approximation ratio.

The idea is to scale all costs such that the above condition holds. The problem is that the straightforward approach of multiplying all costs with the minimum c_{ij} or f_i might overly blow up the coefficient ρ, or it may even be infeasible for zero valued costs. For that reason, we need to perform a more subtle scaling that is inspired by a similar technique given in [3].

The parameter ρ is a lower bound for the objective value μ_{OPT} of the optimal solution OPT. Because there are n clients, all stars with cost-efficiency smaller than ρ/n can be added to a solution ALG, incurring costs at most μ_{OPT}. This observation motivates the following scaling procedure performed at the beginning of the algorithm.

1. For every facility i, choose the largest set B_i of clients such that $(f_i + \sum_{j\in B_i} c_{ij})/|B_i| \leq \rho/n$. If such a B_i exists, set $y_i := 1$ and $x_{ij} := 1$ for all $j \in B_i$. Let C' and F' be the set of unconnected clients and unpicked facilities, respectively.

2. For all $j \in C'$ and $i \in F'$, set $c'_{ij} := nc_{ij}/\rho$ and $f'_i := nf_i/\rho$, respectively. Clients in $C \setminus C'$ do not participate in the algorithm further.

3. Execute Algorithms 1 and 2 with clients and facilities in C' and F, the coefficient $\rho' = n$ (the new ρ), and costs c'_{ij} and f'_i.

Notice that the above procedure can be executed in our distributed model in a constant number of communication rounds. The facility location instance resulting from the above transformation fulfils the following useful and simple property.

Lemma 7.1. *Consider a facility location instance derived from the above transformation. Throughout the algorithm and for all $i \in F$, it holds that $c(i) \geq 1$.*

Proof. If B is the set of clients constituting $c(i)$, then

$$c(i) = \frac{f'_i + \sum_{j\in B} c'_{ij}}{|B|} = \frac{\frac{n}{\rho}\left(f_i + \sum_{j\in B} c_{ij}\right)}{|B|}$$

Assume for contradiction that $i \in F$ and $c(i) < 1$. It follows that $\frac{f_i + \sum_{j\in B} c_{ij}}{|B|} < \frac{\rho}{n}$. This contradicts the fact that the star B was *not* selected during the transformation, that is, the clients $j \in B$ are in C'. □

Having Lemma 7.1 allows us to use Lemmas 5.4 and 5.9 as in the proof of Section 5. The remainder of the proof in Section 5 remains the same.

Summarizing, the transformation algorithm runs in $O(k^2)$ rounds and yields a solution of cost at most $O(k(mn)^{1/k}) \cdot \mu_{OPT} + \mu_{OPT}$ for the fractional facility location problem. In combination with randomized rounding, this results in a $O(k(mn)^{1/k}\log(n+m))$ approximation.

8. FAULT-TOLERANCE

Facility location problems model the tradeoff between the cost of developing resources and the utility accruing from them. In numerous applications, fault-tolerance is of importance in this context [28]. When considering caching in a network, for instance, the caches should be resistent to failures of nodes and links. This fault-tolerance is modelled by demanding that every client j be assigned to at least r_j facilities, r_j being the *requirement* of client j. The special case $r_j = 1, \forall j \in C$ corresponds to the regular facility location problem. The fault-tolerant facility location problem can be captured by the following LP relaxation.

$$\begin{array}{rrll} \min & \sum_{i\in F} f_i y_i & + \sum_{i\in F}\sum_{j\in C} c_{ij}x_{ij} & \\ & \sum_{i\in F} x_{ij} & \geq r_j & , \forall j \in C \\ & y_i - x_{ij} & \geq 0 & , \forall j \in C, i \in F \\ & -x_{ij} & \geq -1 & , \forall j \in C, i \in F \\ & x_{ij}, y_i & \geq 0 & , \forall j \in C, i \in F \end{array}$$

The additional constraint $x_{ij} \leq 1$ prevents a client from connecting to the same facility multiple times.

The algorithm in Section 4 can be adapted for the fault-tolerant case. Analogously to the regular case, the coefficient ρ' is defined as the minimal cost such that every city can open and connect to k facilities. The algorithm is changed such that, clients remain in the active set $\mathcal{A}$ until they are (fractionally) covered by *at least* r_j facilities.

9. CONCLUSIONS

Many applications of the facility location problem such as caching in the Internet inherently apply to distributed settings. In this paper, we have given a classification of the trade-off between the amount of communication and the quality of the obtained global solution. Our solution technique is based on the distributed approximation of a linear program which is, in contrast to previous work [23, 3, 17], not a covering or packing problem. By thus pushing the boundaries of distributed LP approximation, we hope that our paper is a step towards understanding the nature of more general linear programs in a distributed context.

Our results give raise to several questions. First, the fact that in the centralized case, the *metric* facility location problem allows constant approximations [27, 13] raises hope for faster approximations algorithms in distributed settings, too. Moreover, our problem setting is a complete bipartite graph. Interestingly, there are virtually no lower bounds for the bounded message size model for graphs with diameter 1 or 2. For instance, all lower bounds for the MST problem apply to graphs with diameter at least 3. Finding lower bounds for this model appears to be an outstanding open problem.

10. ACKNOWLEDGEMENTS

We would like to thank the anonymous PODC reviewers for pointing out a simplification in the original version of Lemma 5.9.

11. REFERENCES

[1] B. Awerbuch. Complexity of Network Synchronization. *Journal of the ACM*, 32(4):804–823, 1985.

[2] M. L. Balinski. On Finding Integer Solutions to Linear Programs. In *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*, pages 225–248, 1966.

[3] Y. Bartal, J. W. Byers, and D. Raz. Global Optimization Using Local Information with Applications to Flow Control. In *Proc. 38th Symposium on Foundations of Computer Science (FOCS)*, pages 303–312, 1997.

[4] G. Cornuejols, G. Nemhauser, and L. Wolsey. *Discrete Location Theory*, chapter The Uncapacitated Facility Location Problem, pages 119–171. Wiley, 1990.

[5] M. Elkin. An Unconditional Lower Bound on the Hardness of Approximation of Distributed Minimum Spanning Tree Problem. In *Proceedings of the 36th annual ACM Symposium on Theory of Computing (STOC)*, pages 331–340, 2004.

[6] M. Elkin. Distributed Approximation - A Survey. *ACM SIGACT News - Distributed Computing Column*, 35(4), 2004.

[7] U. Feige. A Threshold of $\ln n$ for Approximating Set Cover. *Journal of the ACM*, 45(4):634–652, 1998.

[8] M. X. Goemans and D. P. Williamson. A General Approximation Technique for Constrained Forest Problems. *SIAM J. Comput.*, 24(2):296–317, 1995.

[9] M. T. Hajiaghayi, M. Mahdian, and V. S. Mirrokni. The Facility Location Problem with General Cost Functions. *Networks*, 42(1):42–47, 2003.

[10] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proc. 33rd Hawaii International Conference on System Sciences (HICSS)*, page 8020, 2000.

[11] D. S. Hochbaum. Heuristics for the fixed cost median problem. *Math. Programming*, 22:148–162, 1982.

[12] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani. Greedy Facility Location Algorithms analyzed using Dual Fitting with Factor-Revealing LP. *Journal of the ACM*, 50(6):795–824, 2003.

[13] K. Jain and V. V. Vazirani. Primal-Dual Approximation Algorithms for Metric Facility Location and k-Median Problems. In *Proceedings of the 40th Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society, 1999.

[14] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a Local Search Heuristic for Facility Location Problems. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1–10. Society for Industrial and Applied Mathematics, 1998.

[15] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Initializing newly deployed ad hoc and sensor networks. In *Proceedings of 10th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 260–274, 2004.

[16] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What Cannot be Computed Locally! In *Proceedings of the 23rd ACM Symposium on the Principles of Distributed Computing (PODC)*, pages 300–309, 2004.

[17] F. Kuhn and R. Wattenhofer. Constant-Time Distributed Dominating Set Approximation. In *Proc. of the 22nd Annual ACM Symp. on Principles of Distributed Computing (PODC)*, pages 25–32, 2003.

[18] J.-H. Lin and J. S. Vitter. ϵ-Approximations with Minimum Packing Constraint Violation. In *Proceedings of the 24th ACM Symposium on Theory of Computing (STOC)*, pages 771–782, 1992.

[19] N. Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[20] Z. Lotker, E. Pavlov, B. Patt-Shamir, and D. Peleg. MST Construction in O(log log n) Communication Rounds. In *Proceedings of the 15th ACM symposium on Parallel Algorithms and Architectures (SPAA)*, pages 94–100. ACM Press, 2003.

[21] M. Luby and N. Nisan. A Parallel Approximation Algorithm for Positive Linear Programming. In *Proc. of the 25th ACM Symposium on Theory of Computing (STOC)*, pages 448–457, 1993.

[22] C. Lund and M. Yannakakis. On the Hardness of Approximating Minimization Problems. *Journal of the ACM*, 41(5).

[23] C. Papadimitriou and M. Yannakakis. Linear Programming Without the Matrix. In *Proceedings of the 25th ACM Symposium on Theory of Computing (STOC)*, pages 121–129. ACM Press, 1993.

[24] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications, 2000.

[25] D. Peleg and V. Rubinovich. A Near-Tight Lower Bound on the Time Complexity of Distributed Minimum-Weight Spanning Tree Construction. *SIAM J. Comput.*, 30(5):1427–1442, 2001.

[26] S. Rajagopalan and V. Vazirani. Primal-Dual RNC Approximation Algorithms for Set Cover and Covering Integer Programs. *SIAM Journal on Computing*, 28:525–540, 1998.

[27] D. B. Shmoys, E. Tardos, and K. I. Aardal. Approximation Algorithms for Facility Location Problems. In *Proc. 29th Symposium on Theory of Computing (STOC)*, pages 265–274, 1997.

[28] C. Swamy and D. B. Shmoys. Fault-Tolerant Facility Location. In *Proc. 14th Symposium on Discrete Algorithms (SODA)*, pages 735–736, 2003.

[29] D. P. Williamson, M. X. Goemans, M. Mihail, and V. V. Vazirani. A Primal-Dual Approximation Algorithm for Generalized Steiner Network Problems. In *Proc. 25th Symposium on Theory of Computing (STOC)*, pages 708–717, 1993.