

# Return of the Primal-Dual: Distributed Metric Facility Location

Saurav Pandit      Sriram Pemmaraju  
Dept. of Comp. Sci, Univ. of Iowa  
Iowa City, IA 52242-1419  
[spandit, sriram]@cs.uiowa.edu

## ABSTRACT

In this paper we present fast, distributed approximation algorithms for the *metric facility location* problem in the *CONGEST* model, where message sizes are bounded by  $O(\log N)$  bits,  $N$  being the network size. We first show how to obtain a 7-approximation in  $O(\log m + \log n)$  rounds via the primal-dual method; here  $m$  is the number of facilities and  $n$  is the number of clients. Subsequently, we generalize this to a  $k$ -round algorithm, that for every constant  $k$ , yields an approximation factor of  $O(m^{2/\sqrt{k}} \cdot n^{3/\sqrt{k}})$ . These results answer a question posed by Moscibroda and Wattenhofer (*PODC 2005*). Our techniques are based on the primal-dual algorithm due to Jain and Vazirani (*JACM 2001*) and a rapid randomized sparsification of graphs due to Gfeller and Vicari (*PODC 2007*). These results complement the results of Moscibroda and Wattenhofer (*PODC 2005*) for *non-metric* facility location and extend the results of Gehweiler et al. (*SPAA 2006*) for uniform metric facility location.

**Categories and Subject Descriptors:** C.2.4 [Computer-Communication Networks]: Distributed Systems

**General Terms:** Algorithms, Performance, Theory.

**Keywords:** Facility Location, Approximation Algorithms, Bounded Message Size, Wireless Ad-hoc Networks, Unit Ball Graphs.

## 1. INTRODUCTION

Recent research in the area of distributed approximation algorithms [4, 6, 10, 11, 12, 13, 14, 17] has led to interesting and sometimes optimal trade-offs between the amount of resources used (e.g., number of communication rounds, size of messages, etc.) and the quality of solution (i.e., approximation factor) obtained. In recent years, such trade-offs have been especially well-studied for the minimum spanning tree problem [10] and the dominating set problem [12, 13, 14]. One theme in this research is motivated by the question: can one design distributed approximation algorithms that provide non-trivial approximation guarantees even when run

for very few (e.g., constant) number of rounds? The earliest example of such an algorithm, as far as we know, is the dominating set algorithm due to Kuhn and Wattenhofer [13] that runs in  $k^2$  rounds, for any  $k$ , and outputs a dominating set whose expected size is within  $O(k^2 \Delta^{2/k} \log \Delta)$  of OPT. Here  $\Delta$  is the maximum degree of the network. In this paper, we investigate this question for the *metric facility location* problem. For instances with  $m$  facilities and  $n$  clients, we first present a 7-approximation algorithm running in  $O(\log m + \log n)$  rounds and then generalize this algorithm to a  $k$ -round algorithm that yields an  $O(m^{2/\sqrt{k}} \cdot n^{3/\sqrt{k}})$ -approximation, for any constant  $k$ . In fact, the  $k$ -round algorithm yields this approximation factor for all  $k = O((\frac{\log m}{\log \log m})^2)$ . A key constraint of our model of distributed computation is that message sizes are bounded above by  $O(\log(m+n))$  bits.

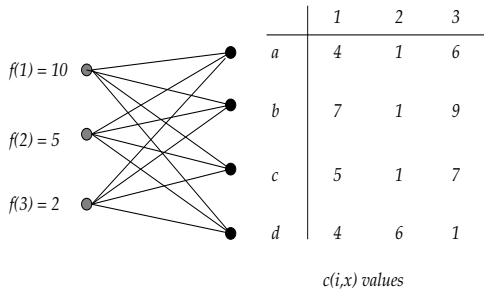
The *facility location* problem takes as input a complete bipartite graph  $G = (F, C, E)$ , where  $F$  is the set of *facilities* and  $C$  is the set of *clients* (or *cities*), facility opening costs  $f : F \rightarrow \mathbb{R}^+$ , and connection costs  $c : E \rightarrow \mathbb{R}^+$ . The goal is to find a set of facilities  $I \subseteq F$  to open and a function  $\phi : C \rightarrow I$  that assigns every client to an open facility so as to minimize  $\sum_{i \in I} f(i) + \sum_{j \in C} c(j, \phi(j))$ . In other words, the goal is to minimize the total cost of opening facilities and connecting clients to open facilities. This is an old and well studied problem in operations research [3], that arises in contexts such as locating hospitals in a city or locating distribution centers in a region. More recently, this problem has also been used as an abstraction for the problem of locating resources in wireless networks [5, 18]. The facility location problem comes in two main versions: the *non-metric* version and the *metric* version. The connection costs in a facility location instance are said to satisfy the *triangle inequality* if for any  $i, i' \in F$  and  $j, j' \in C$ ,  $c(i, j) \leq c(i, j') + c(i', j) + c(i', j)$ . In the *metric facility location* problem the connection costs satisfy the triangle inequality; when they don't, we have the more general *non-metric facility location* problem. See Figure 1 for an illustration. This distinction is important from an approximation point of view because there are a number of sequential constant-factor approximation algorithms for the metric facility location problem ([2, 9, 20] are some examples), whereas for the non-metric facility location problem, the best known approximation factor is  $O(\log n)$  and this is optimal [8, 15].

Our work in this paper is partly motivated by the results of Moscibroda and Wattenhofer [17] who consider the *non-metric* facility location problem and design an approximation algorithm that in  $O(k)$  rounds, for any constant  $k$ , yields a non-trivial approximation factor that depends on  $k$

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PODC'09, August 10-12, 2009, Calgary, Alberta, Canada.

Copyright 2009 ACM 978-1-60558-396-9/09/08 ...\$5.00.



**Figure 1: An instance of the facility location problem.** The table shows the pairwise connection costs between clients and facilities. OPT consists of open facilities 2 and 3 with clients  $a$ ,  $b$  and  $c$  connected to facility 2 and client  $d$  to facility 3. The total cost of OPT is 11. Note that any solution with a single open facility or with all the facilities open, will have cost more than 11. So is the case for any solution that opens facility 1.

and the input size. The underlying network on which this algorithm runs is the complete bipartite graph  $G = (F, C, E)$ , induced by the facilities and the clients. Given that the network has diameter 2, the problem is trivial (i.e., simply reduces to the sequential setting) if arbitrary amounts of information are allowed to be exchanged between neighbors in each round. Placing a reasonable bound, such as  $O(\log(m+n))$ , on the message sizes makes the problem challenging and allows Moscibroda and Wattenhofer [17] to highlight an interesting trade-off between the number of communication rounds of the algorithm and the approximation factor achieved. Specifically, they present an algorithm that runs in  $O(k)$  rounds and yields an approximation of  $O(\sqrt{k}(m\rho)^{1/\sqrt{k}} \log(m+n))$ , where  $m$  is the number of facilities,  $n$  is the number of clients, and  $\rho$  is a coefficient that depends on the numbers in the given instance (i.e., the opening costs and the connection costs). To focus more clearly on the growth of this approximation factor as a function of  $k$ , let us assume that  $\rho = m = n$ . Assuming that  $\rho$  is equal to  $m$  or  $n$  is reasonable since Moscibroda and Wattenhofer [17] assume that  $\rho$  can be stored in a message of size  $O(\log n)$  bits, implying that the value of  $\rho$  is bounded above by a polynomial in  $n$ . With this simplification we see that the Moscibroda-Wattenhofer algorithm runs in  $O(k)$  rounds yielding an approximation factor of  $O(\sqrt{k}n^{2/\sqrt{k}} \log n)$ . Since the metric facility location problem is a special case of the non-metric problem, this leads to question of whether it is possible to design  $k$ -round algorithms for *metric* problem that give a better approximation factor.

This question is motivated by the expectation that in a distributed setting such as this, in which network diameter is not a concern (since it is just a constant), one should be able to obtain, in polylogarithmic number of rounds, approximation factors that are as good as the best approximation factors possible in a sequential setting. Since the metric facility location problem has a sequential constant-factor approximation, ideally we should be able to obtain an approximation factor  $f(m, n, k)$  that tends to a constant as  $k$  tends to some polylogarithmic quantity in  $m$  and  $n$ . But the Moscibroda-Wattenhofer algorithm [17] falls short in the re-

gard of achieving a constant factor approximation, because no matter how large we make  $k$ , the smallest value achieved by their approximation factor of  $O(\sqrt{k}n^{2/\sqrt{k}} \log n)$  is just  $O(\log^2 n)$ . At a high level, the reason for this is that their algorithm [17] uses a 2-step approach that involves solving an LP-relaxation in the first step and doing an independent randomized rounding of the fractional solution, in the second step; unfortunately, in both of these steps there is a logarithmic “overhead” in the approximation factor. These overheads seem fairly fundamental to their approach and to do better, it seems that different techniques are needed.

**Our results.** Our algorithms use a model of distributed computing that is similar to the Moscibroda-Wattenhofer model [17]. Specifically, we assume that the underlying network is the complete bipartite graph  $G = (F, C, E)$  induced by facilities and clients. Also, the network executes algorithms synchronously and in each round nodes receive messages from neighbors, perform polynomial-time local computations, and send  $O(\log(m+n))$ -sized messages to neighbors. This message passing model with a logarithmic bound (in the network size) on the message size has been called the *CONGEST* model by Peleg [19]. The *CONGEST* model is a “point-to-point” communication model and allows a node to send different messages to each of its neighbors, but in our algorithm it suffices for each node to send the same message to *all* of its neighbors. This is the *local broadcast* model that is commonly used in distributed algorithms on wireless networks. To focus more clearly of the challenges of using the *CONGEST* model, we assume that each piece of information that a node possesses (IDs,  $f_i$ ’s,  $c_{ij}$ ’s, etc.) can fit into an  $O(\log(m+n))$ -bit message. We consider the metric facility location problem in this model and present two results: (i) a 7-approximation running in  $O(\log m + \log n)$  rounds and (ii) a  $k$ -round algorithm that yields an  $O(m^{2/\sqrt{k}} \cdot n^{3/\sqrt{k}})$ -approximation. These results answer a question on fast distributed approximation algorithms for the metric facility location problem, posed by Moscibroda and Wattenhofer [17]. Our techniques are based on the Jain-Vazirani primal-dual algorithm [9] and a rapid, distributed, randomized algorithm for sparsification of graphs due to Gfeller and Vicari [7]. In fact the success of the primal-dual approach in this setting, in a sense, contradicts the claim by Moscibroda and Wattenhofer [17] that the Jain-Vazirani primal-dual approach works in the distributed setting only if “either message-size is unbounded, or the algorithm’s time-complexity depends on the size of the problem instance.” This claim is true only if maximal independent set (MIS) computation is viewed as an indispensable part of the Jain-Vazirani algorithm. We obtain our results by replacing the MIS computation step with a much faster sparsification step which makes sufficient progress in constant number of rounds, even with bounded message size.

Another motivation for our work comes from the elegant *constant-round*, constant-factor approximation algorithm for the *uniform* metric facility location problem by Gehweiler et al. [6]. The adjective “uniform” in the title of the problem refers to the assumption that all facility opening costs are identical. The Gehweiler et al. [6] algorithm also assumes the *CONGEST* model and further assumes that the underlying network is a clique, with each node being a client as well as a possible location for a facility. The results in this paper are partly motivated by the desire to extend the Gehweiler et al. algorithm [6] to instances that allow arbitrary facility

opening costs. While we describe our algorithm for a bipartite network, the algorithm works more or less unchanged, in the clique setting of Gehweiler et al. [6].

## 2. LOGARITHMIC-ROUND ALGORITHM

For the rest of the paper, let  $m = |F|$ ,  $n = |C|$ , and  $N = m + n$ . This section is devoted to our  $O(\log n + \log m)$ -round, 7-approximation algorithm. This algorithm will set the stage for the more complicated  $k$ -round approximation algorithm described in the next section. We assume that each message can be  $O(\log N)$  bits long and that all node IDs can be represented in  $\log N$  bits. Note that in a complete bipartite graph, every node can learn the total number of facilities and clients in two rounds of communication. So for all algorithms described in this paper, we assume that all nodes know  $m$  and  $n$ .

### 2.1 Background

Our algorithm is based on the primal-dual algorithm for facility location due to Jain and Vazirani [9] which we briefly review here. The starting point of this algorithm is the following Integer Program (IP) representation of facility location. Here  $y_i$  indicates whether facility  $i$  is open and  $x_{ij}$  indicates if client  $j$  is connected to facility  $i$ . The first set of constraints ensure that every client is connected to a facility and the second set of constraints guarantee that each client is connected to an open facility.

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in F, j \in C} c_{ij} \cdot x_{ij} + \sum_{i \in F} f_i \cdot y_i \\ \text{subject to} \quad & \sum_{i \in F} x_{ij} \geq 1, & j \in C \\ & y_i - x_{ij} \geq 0, & i \in F, j \in C \\ & x_{ij} \in \{0, 1\}, & i \in F, j \in C \\ & y_i \in \{0, 1\}, & i \in F \end{aligned}$$

As is standard, we work with the LP-relaxation (LP) of the above IP obtained by replacing the integrality constraints by  $x_{ij} \geq 0$  for all  $i \in F$ ,  $j \in C$  and  $y_i \geq 0$  for all  $i \in F$ . The dual (DP) of this LP-relaxation is the following:

$$\begin{aligned} \text{maximize} \quad & \sum_{j \in C} \alpha_j \\ \text{subject to} \quad & \alpha_j - \beta_{ij} \leq c_{ij}, & i \in F, j \in C \\ & \sum_{j \in C} \beta_{ij} \leq f_i, & i \in F \\ & \alpha_j \geq 0, & j \in C \\ & \beta_{ij} \geq 0, & i \in F, j \in C \end{aligned}$$

The dual variable  $\alpha_j$  can be interpreted as the amount that client  $j$  is willing to pay in order to connect to a facility. Of this amount,  $c_{ij}$  goes towards paying for connecting to facility  $i$ , whereas the “extra”, namely  $\beta_{ij}$ , is seen as the payment of client  $j$  towards opening facility  $i$ . The goal is to obtain an integral feasible solution  $\{x_{ij}, y_i\}$  to the LP and a feasible solution  $\{\alpha_j, \beta_{ij}\}$  to the DP such that

$$\sum_{i \in F, j \in C} c_{ij} \cdot x_{ij} + \sum_{i \in F} f_i \cdot y_i \leq c \cdot \left( \sum_{j \in C} \alpha_j \right),$$

for some constant  $c$  and by the weak duality theorem, this guarantees that  $\{x_{ij}, y_i\}$  is a  $c$ -approximation to the facility location problem.

## 2.2 Overview of Algorithm

The algorithm consists of three phases, each of which is overviewed in Algorithm 2.2. At a high level this algorithm

---

### Algorithm 1 Algorithm overview

---

- 1: **Initialization phase** (runs in constant rounds). We identify *low-paying* clients and *cheap* facilities. Cheap facilities are opened and low-paying clients are connected to cheap facilities. Initial  $\alpha_j$ -values for the remaining (i.e., non-low-paying) clients are computed.
  - 2: **Primal-Dual phase** (runs in  $O(\log n)$  rounds). Each client  $j$ , that is not low-paying, increments its dual variable (i.e.,  $\alpha_j$ ) geometrically in each round until a facility  $i$  is found such that the connection cost  $c_{ij}$  is “paid for” by  $\alpha_j$  and the facility opening cost  $f_i$  is paid for by the payments of some clients. Facility  $i$  is declared *temporarily open* and  $j$  is connected to  $i$ .
  - 3: **Sparsification phase** (runs in  $O(\log m)$  rounds). We use Luby’s MIS algorithm [16] to shut down some of the temporarily open facilities and declare the rest permanently opened. Some clients are reconnected as the result of the partial shut down.
- 

is quite similar to the Jain-Vazirani algorithm, but several challenges have to be dealt with in order for the algorithm to run in logarithmic number of rounds in the *CONGEST* model. For example, we have to quickly find an initial feasible solution to the DP that is not too far away from the “final” feasible solution so that the Primal-Dual Phase can terminate in  $O(\log n)$  rounds.

### 2.3 Initialization Phase

Set  $\alpha_j := \frac{1}{n} \min_i (f_i + c_{ij})$  for all  $j \in C$ . For each  $j \in C$ , let  $i^*$  be such that  $f_{i^*} + c_{i^*j} = \min_i (f_i + c_{ij})$ , and set  $\phi(j) := i^*$ . For each  $i \in F$ ,  $j \in C$ , set  $\beta_{ij} := \max\{\alpha_j - c_{ij}, 0\}$ .

LEMMA 1. *The set of  $\alpha_j$ -values and  $\beta_{ij}$ -values defined above constitute a feasible solution to the DP.*

PROOF. It is easy to verify the constraints  $\alpha_j - \beta_{ij} \leq c_{ij} \forall i \in F, \forall j \in C$ ,  $\alpha_j \geq 0 \forall j \in F$  and  $\beta_{ij} \geq 0 \forall i \in F, \forall j \in C$ . Hence, we will focus on the constraints  $\sum_{j \in C} \beta_{ij} \leq f_i, \forall i \in F$ .

Let us fix an  $i \in F$  and examine  $\beta_{ij}$  for each  $j \in C$ . If  $\beta_{ij} = 0$ , it is clear that  $\beta_{ij} - f_i \leq 0$  and more to the point,

$$\beta_{ij} \leq \frac{f_i}{n} \tag{1}$$

On the other hand, if  $\beta_{ij} > 0$ , then it must be the case that  $\beta_{ij} = \alpha_j - c_{ij}$ . For this  $j$ , suppose that  $\phi(j) = i^*$ , i.e.,

$$\alpha_j = \frac{1}{n} \left( \min_i (f_i + c_{ij}) \right) = \frac{1}{n} (f_{i^*} + c_{i^*j}).$$

Hence,  $\beta_{ij} = \frac{1}{n} (f_{i^*} + c_{i^*j} - nc_{ij})$ . By choice of  $i^*$ ,

$$\begin{aligned} \beta_{ij} - \frac{f_i}{n} &= \frac{1}{n} \left( f_{i^*} + c_{i^*j} - nc_{ij} - f_i \right) \\ &= \frac{1}{n} \left( (f_{i^*} + c_{i^*j}) - (f_i + c_{ij}) - (n-1)c_{ij} \right) \\ &\leq 0. \end{aligned}$$

Simply put, for the arbitrary  $i$  we fixed and  $\forall j \in C$

$$\beta_{ij} - \frac{f_i}{n} \leq 0. \tag{2}$$

Summing over all  $j$ , using inequalities (1) and (2), we get  $\sum_{j \in C} \beta_{ij} - f_i \leq 0$ .  $\square$

Now, let us define  $\alpha^* = \max_j \alpha_j$  and let  $j^* = \arg \max \alpha_j$ . Any  $j \in C$  with  $\alpha_j \leq \alpha^*/n^2$  will be called a *low-paying* client and the facility  $\phi(j)$  (for low-paying  $j$ ) will be called a *cheap* facility. The reason for identifying these low-paying clients and cheap facilities is that their overall opening and connection cost is so low that they can be dealt without much further ado (so shown in the next lemma). Furthermore, eliminating these clients is critical in guaranteeing that the next phase (i.e., the primal-dual phase) runs in  $O(\log n)$  rounds. In preparation for the next lemma, let  $L$  be the set of low-paying clients and  $OPT$  be the cost of an optimal solution to the facility location problem.

**LEMMA 2.** *The total cost of opening all cheap facilities and connecting each low-paying client  $j$  to facility  $\phi(j)$  is at most  $OPT$ .*

**PROOF.** The total cost of opening the cheap facilities and connecting each low-paying client  $j$  to cheap facility  $\phi(j)$  is bounded above by

$$\sum_{j \in L} (c_{\phi(j)j} + f_{\phi(j)}) = \sum_{j \in L} n \cdot \alpha_j \leq \sum_{j \in L} \frac{\alpha^*}{n}.$$

Since the total number of clients is  $n$ , the right hand side above is bounded above by  $\alpha^*$ . Note that there is a client  $j^*$  is such that  $\alpha_{j^*} = \alpha^*$  and furthermore the  $\alpha_j$ 's are all part of a feasible solution to the DP. Therefore, by weak duality  $\alpha^* \leq \sum_{j \in C} \alpha_j \leq OPT$ .  $\square$

The aforementioned process of identifying the low-paying clients and cheap facilities can be accomplished quite easily in just  $O(1)$  rounds of communication with messages of size  $O(\log N)$ .

## 2.4 The Primal-Dual Phase

Here we describe the primal-dual phase of the algorithm in which  $\alpha_j$ 's are geometrically raised in a synchronous manner and a set of facilities are “temporarily” opened. To prevent low-paying clients and cheap facilities from getting in the way of the primal-dual phase, for each low-paying client  $j$ , we set  $color(j) := grey$  and  $status(\phi(j)) := open$ . The *color* of a client  $j$  will indicate to the primal-dual phase whether  $j$  is still active or not; the color *grey* will denote inactivity (i.e.,  $\alpha_j$  is not raised any more). The *status* of a facility can be one of  $\{closed, temporarily-open, open\}$  and the cheap facilities are all declared open and do not participate in the primal-dual phase.

Recalling that  $L$  denotes the set of low-paying facilities, let  $\alpha_{min} := \min_{j \notin L} \alpha_j$ . Thus  $\alpha_{min}$  is the smallest  $\alpha_j$  value, excluding the “very small”  $\alpha_j$  values belonging to the low-paying clients. Also, note that  $\alpha^*/n^2 < \alpha_{min} \leq \alpha^*$ .

We next describe a typical iteration (say, iteration  $s$ ) of Client  $j$ 's Primal-Dual Algorithm (see Algorithm 2). Initially, every client  $j$  (that is not low-paying) sets its  $\alpha_j$ -value to  $\alpha_{min}$  and sets its *color* to *white* indicating that it is ready to be active. The clients increase their  $\alpha_j$ -values synchronously, in every iteration, by a constant multiplicative factor (2 in this case). These  $\alpha_j$ -values should be viewed

---

### Algorithm 2 Primal-Dual Client $j$ Algorithm

---

```

1: Init: If  $j$  is not a low-paying client:  $\alpha_j \leftarrow \alpha_{min}$ ;
    $color(j) \leftarrow white$ .
2: {Activity in each iteration:}
3: if ( $color(j) = white$ ) then
4:    $Send[\alpha_j]$ 
5:    $\forall i: \gamma_{ij} \leftarrow \alpha_j - c_{ij}$ 
6:    $\forall i: Receive[status(i)]$ 
7:   if ( $\exists i$  such that
   ( $status(i) \in \{temporarily-open, open\} \wedge \gamma_{ij} \geq 0$ ))
   then
8:      $color(j) \leftarrow grey$ 
9:      $\phi(j) \leftarrow i$ 
10:  end if
11:   $Send[color(j)]$ 
12:   $\alpha_j \leftarrow 2 \cdot \alpha_j$ 
13: end if

```

---

as “payments” and the payment  $\alpha_j$  of client  $j$  first goes towards paying off the connection costs  $c_{ij}$  for all  $i \in F$ . The sign of  $\gamma_{ij}$  (Line 5) indicates whether  $j$  has paid for a connection cost  $c_{ij}$  or not. If, during an iteration,  $j$  discovers that it has paid enough towards a connection cost  $c_{ij}$ , then there are two possibilities: (i)  $i$  is already open, in which case the client  $j$  is connected to  $i$ , by setting  $\phi(j)$  to  $i$  (Line 9) and (ii)  $i$  is not open and in this case  $j$ 's payments will go towards opening the facility  $i$ , i.e., towards  $\beta_{ij}$ . The *color* of a client denotes whether it has been connected or not. Initially all active clients are *white*. Each client changes its own color to *grey* as soon as they get connected.

All facilities that are not cheap, are initialized to have a *closed status*. During the course of a typical iteration of facility  $i$ 's algorithm (shown in Algorithm 3), the facility receives payments  $\alpha_j$  from clients, determines their residual payments  $\beta_{ij}$  (after accounting for payment towards connection costs), and checks if the residual payments  $\beta_{ij}$ , for all  $j \in C$  are sufficient to pay for the cost of opening  $f_i$  (in Line 6). When the opening cost of a facility is completely paid for, it sets its status to *temporarily-open*.

---

### Algorithm 3 Primal-Dual Facility $i$ Algorithm

---

```

1: Init:  $status(i) \leftarrow closed$  if  $i$  is not a cheap facility
2: {Activity in each iteration:}
3: if ( $status(j) = closed$ ) then
4:    $\forall j: Receive[\alpha_j]$ 
5:    $\forall j$  such that  $color(j) = white: \beta_{ij} \leftarrow \max\{\alpha_j - c_{ij}, 0\}$ 
6:   if ( $\sum_{j \in C} \beta_{ij} \geq f_i$ ) then
7:      $status(i) \leftarrow temporarily-open$ 
8:   end if
9:    $Send[status(i)]$ 
10:   $\forall j: Receive[color(j)]$ 
11: end if

```

---

Note that, corresponding to the three message transactions (*Send - Receive - Send*) in a typical iteration of a client's algorithm, there are three message transactions (*Receive - Send - Receive*) in a facility's algorithm. We assume that these transactions take place in a synchronous manner. All activities in the primal-dual algorithm cease when all clients

become *grey*. This is because when clients become *grey* they no longer increase their  $\alpha_j$ -values and without any change in the  $\alpha_j$ -values no new facilities will be paid for. We now prove two important properties of the Primal-Dual Algorithm.

LEMMA 3. *The Primal-Dual Algorithm runs for at most  $9 \log n$  communication rounds.*

PROOF. For client  $j$ , let  $\alpha_j^{(s)}$  denote the value of  $\alpha_j$  after iteration  $s$ . We know  $\alpha_j^{(0)} = \alpha_{min}$  and also  $\alpha_{min} > \alpha^*/n^2$ . Since  $\alpha_j$ 's double in each iteration,  $\alpha_j^{(t)} > n \cdot \alpha^*$  for  $t := 3 \log n$ . Now recall from the initialization phase that  $\alpha^* = \max_j \min_i (f_i + c_{ij})/n$ . Therefore,  $\alpha_j^{(t)}$  will exceed  $\min_i (f_i + c_{ij})$ , which means that client  $j$ , all by itself, is paying for more than the amount needed to open facility  $i^*$ , where  $i^*$  is  $\operatorname{argmin}_i (f_i + c_{ij})$ . Therefore client  $j$ , for all  $j \in C$ , turns grey in round  $t = 3 \log n$  or earlier. Each iteration of the primal-dual algorithms corresponds to 3 communication rounds, yielding the result.  $\square$

LEMMA 4. *After the Primal-Dual Algorithm has terminated,  $\sum_{j \notin L} \alpha_j \leq 2 \cdot OPT$ .*

PROOF. Set  $\bar{\alpha}_j = 0$  for all  $j \in L$  and  $\bar{\alpha}_j = \alpha_j/2$  for all  $j \notin L$ . Set  $\bar{\beta}_{ij} := \max\{c_{ij} - \bar{\alpha}_j, 0\}$  for all  $i \in F, j \in C$ . It is easy to check that this is a feasible solution to the DP and therefore by weak duality  $\sum_{j \in C} \bar{\alpha}_j \leq OPT$ . This yields the claim.  $\square$

## 2.5 The Sparsification Phase

At the end of the Primal-Dual Phase we have a solution (albeit, a costly one, possibly) to the facility location problem, given by the open and temporarily open facilities and each client connecting to an open or temporarily open facility. The goal of the sparsification phase is to obtain a cheaper solution that can be “charged” to the dual variables and since the sum of the dual variables is within a constant times  $OPT$  (Lemma 4), we will get a constant-factor approximation. This is similar to the last phase in the Jain-Vazirani algorithm [9]; our contribution here is to show that it can be implemented in  $O(\log m)$  rounds in the  $CONGEST$  model.

Let  $F_t$  be the set of temporarily open facilities at the end of the Primal-Dual Phase. Define a graph  $H = (F_t, E_t)$ , where  $E_t$  consists of edges connecting pairs of temporarily open facilities  $i$  and  $i'$  for which there is a client  $j$  such that  $\beta_{ij} > 0$  and  $\beta_{i'j} > 0$ ; in other words,  $j$  makes a positive payment towards the opening of both  $i$  and  $i'$ . The high level algorithm for the sparsification phase is shown in Algorithm 4.

Note that since  $M$  is an MIS of  $H$ , for every client  $j$  that finds out that the facility  $\phi(j)$  is now closed, there is another facility  $i' \in M$ , that is a neighbor of  $\phi(j)$  in  $H$ , to which  $j$  can be “reconnected.” If a client  $j$  retains its connection from the Primal-Dual Phase, we say that  $j$  is *directly connected* to  $\phi(j)$ ; otherwise, if  $j$  is reconnected in the Sparsification Phase, we say that  $j$  is *indirectly connected* to  $\phi(j)$ .

In the Sparsification Phase, Steps 2 and beyond are all easy to implement in constant number of communication rounds in the  $CONGEST$  model. So we focus on Step 1. The difficulty with using Luby’s algorithm [1, 16] “as is” is

---

### Algorithm 4 Sparsification Phase

---

- 1: Compute a maximal independent set (MIS)  $M$  of  $H$
  - 2: Permanently open each facility in  $M$ , i.e.,  $\forall i \in M : \text{status}(i) \leftarrow \text{open}$
  - 3: Close each facility in  $F_t \setminus M$ , i.e.,  $\forall i \in F_t \setminus M : \text{status}(i) \leftarrow \text{closed}$
  - 4: **for all** clients  $j \in C$  **do**
  - 5:   **if** ( $\text{status}(\phi(j)) = \text{closed}$ ) **then**
  - 6:      $\phi(j) \leftarrow i'$ , where  $i' \in M$  and is a neighbor in  $H$  of  $\phi(j)$
  - 7:   **end if**
  - 8: **end for**
- 

to compute the MIS of  $H$  is that nodes in  $H$  do not have edges to each other in the underlying network and will have to communicate via clients. Given the restriction on the message sizes, this may be hard to do in constant rounds. For example, it is impossible for a node  $i \in F_t$  to quickly (i.e., in constant or even polylogarithmic number of rounds) find out the IDs of all its neighbors in  $H$  since all this information may have to arrive at  $i$  via a single client. Consider a typical stage of Luby’s algorithm:

1. Each, as yet undecided node  $i \in F_t$  marks itself with probability  $1/\text{degree}(i)$ . Here  $\text{degree}(i)$  refers to the degree of  $i$  in the subgraph of  $H$  induced by the undecided vertices.
2. Each node  $i$  that is marked in Step 1 unmarks itself if it finds a neighbor  $j$  with lower degree that has marked itself. Ties can be resolved via the use of IDs.

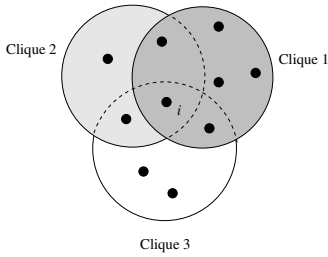
The nodes that mark themselves in a stage of Luby’s algorithm have “decided” to join the MIS and neighbors of such nodes have “decided” not to join the MIS; the undecided nodes continue to the next phase. For Step 1, node  $i \in F_t$  needs to know its degree in the subgraph of  $H$  induced by currently undecided nodes and for Step 2 node  $i$  needs to know the lexicographically smallest tuple  $(ID(i'), \text{degree}(i'))$  over all marked nodes  $i'$  in its neighborhood. Both pieces of information can be computed in constant number of rounds in the  $CONGEST$  model and to do this we take advantage of the fact that  $H$  is essentially composed of a number of interacting cliques, with each clique being “controlled” by a client (see Figure 2). In the interest of space conservation, we skip the details.

## 2.6 Analysis

We now analyze the quality of the computed solution as well as the number of rounds it takes to compute the solution in the  $CONGEST$  model. We start off with a lemma showing that the cost of reconnecting  $j$  to an open facility in the Sparsification Phase can be charged to  $\alpha_j$ . A lemma such as this appears in Jain and Vazirani’s analysis [9] as well; our proof (skipped to save space) is slightly different since events in our algorithm happen in discrete rounds whereas the Jain-Vazirani algorithm grows dual variables continuously.

LEMMA 5. *If  $j$  is indirectly connected to  $i$ , then  $c_{ij} \leq 3\alpha_j$ .*

The following lemma can be proved by routine accounting and we skip its proof as well.



**Figure 2:** Suppose that a temporarily open facility  $i$  is positively paid for by Clients 1, 2 and 3 (i.e.,  $\beta_{i1}, \beta_{i2}, \beta_{i3} > 0$ ). Further suppose that client 1 positively pays for 6 temporarily open facilities, client 2 pays for 4, and client 3 for 5 temporarily open facilities. This leads to 3 cliques in  $H$  of sizes 6, 4, and 5 respectively. If each facility  $i$  thinks of itself as belonging to the clique of the client with lowest ID, then Clique 1 will have size 6, Clique 2 will have size 2, and Clique 3 will have size 2. If the clients keep track of the clique sizes in this manner,  $i$  can correctly figure out its degree, which is  $6+2+2 = 10$ .

LEMMA 6. *The algorithm presented above terminates in  $O(\log m + \log n)$  rounds and uses messages of size  $O(\log N)$  bits.*

Our final lemma shows that our solution is a 7-approximation.

LEMMA 7. *Let  $I$  be the set of facilities opened by our algorithm. Then,*

$$\sum_{i \in I} f_i + \sum_{j \in C} c_{\phi(j)j} \leq 7 \cdot OPT.$$

PROOF. Let  $I' \subseteq I$  denote the facilities that are not cheap and were therefore opened in the Primal-Dual Phase. Let  $COST$  denote the total cost of opening facilities in  $I'$  and the connection cost of clients that are not low-paying. Clients that are not low-paying can be partitioned into sets,  $A$ , consisting of clients that directly connect to a facility and  $B$ , consisting of clients that indirectly connect to a facility. Note that for each  $i \in I'$ ,  $f_i \leq \sum_{j \in A} \beta_{ij}$ . Therefore,

$$\begin{aligned} COST &= \sum_{i \in I'} f_i + \sum_{j \in A} c_{\phi(j)j} + \sum_{j \in B} c_{\phi(j)j} \\ &\leq \sum_{i \in I'} \sum_{j \in A} \beta_{ij} + \sum_{j \in A} (\alpha_j - \beta_{\phi(j)j}) + 3 \sum_{j \in B} \alpha_j \quad (3) \\ &\leq \sum_{j \in A} \beta_{\phi(j)j} + \sum_{j \in A} (\alpha_j - \beta_{\phi(j)j}) + 3 \sum_{j \in B} \alpha_j \quad (4) \\ &\leq \sum_{j \in A} \alpha_j + 3 \sum_{j \in B} \alpha_j \\ &\leq 3 \sum_{j \notin L} \alpha_j \leq 6 \cdot OPT \quad (5) \end{aligned}$$

Inequality (3) is obtained by using the above mentioned upper bound on  $f_i$  where  $i \in I'$ , Lemma 5 and the fact that if client  $j$  is directly connected to  $\phi(j)$  (i.e.  $\forall j \in A$ ),  $\alpha_j \geq c_{\phi(j)j}$  and so  $c_{\phi(j)j} = \alpha_j - \beta_{\phi(j)j}$ . Inequality (4) is obtained by rewriting  $\sum_{i \in I'} \sum_{j \in A} \beta_{ij}$  as  $\sum_{j \in A} \sum_{i \in I'} \beta_{ij}$  and then noting that for each  $j \in A$ ,  $\beta_{ij} > 0$  for exactly one  $i$ , namely  $i = \phi(j)$ . This is the result of the Sparsification Phase. The first part of Inequality (5) follows from the fact

that  $A \cup B$  is the set of clients that are not low-paying. Earlier we had used  $L$  to denote the set of low-paying clients. The second part of this inequality follows from Lemma 4.

Using the above result and Lemma 2, the total cost of the solution, including opening cost of cheap facilities and connection cost of low-paying clients, is  $7 \cdot OPT$ .  $\square$

### 3. THE $k$ -ROUND ALGORITHM

Motivated by the goal of achieving a non-trivial approximation ratio even while using a very small (e.g., a constant) number of rounds, we present in this section a generalization of the Logarithmic-round algorithm. For two tunable, positive, integer parameters  $k_1$  and  $k_2$ , we run the algorithm for  $k := k_1 \cdot k_2$  rounds to obtain an  $O(n^{3/k_1} \cdot m^{2/k_2})$ -approximation algorithm. Recall that  $m$  is the number of facilities and  $n$  is the number of clients. Here  $k_1$  can take any value, whereas  $k_2$  can take any value bounded by  $O(\frac{\log m}{\log \log m})$ . The bound on  $k_2$  comes from our randomized Sparsification Algorithm, which does not exhibit the desired behavior for larger values of  $k_2$ . This is because for larger  $k_2$  the degree of the graph being sparsified falls below  $\log m$  and at that point it is no longer possible to prove properties of the Sparsification Algorithm with high probability.

The Initialization Phase is identical to the one used in the Logarithmic-round algorithm. However, the other two phases (especially the Sparsification Phase) are quite different and furthermore, how the Primal-Dual Phase and the Sparsification Phase interact is also quite different. Rather than running the Sparsification Phase after the Primal-Dual Phase is completed, we now interleave the iterations of the two phases, running  $k_2$  iterations of a Sparsification Algorithm after each iteration of the Primal-Dual Algorithm.

To have the Primal-Dual Algorithm make sufficient progress in each iteration, client  $j$  raises the  $\alpha_j$ -value by a multiplicative factor of  $n^{3/k_1}$ . Recall from Section 2.4 that each  $\alpha_j$  for  $j \notin L$  is initialized to  $\alpha_{min}$ , where  $\alpha_{min} \cdot n^3 > \min_i (f_i + c_{ij})$  for every client  $j$ . Therefore, if client  $j$  were to run  $k_1$  iterations of the Primal-Dual Algorithm, then its  $\alpha_j$  value would grow from  $\alpha_{min}$  to  $\alpha_{min} \cdot (n^{3/k_1})^{k_1} = \alpha_{min} \cdot n^3 > \min_i (f_i + c_{ij})$ . At this point the  $\alpha_j$ -value is more than sufficient to pay for the opening cost of a facility all by itself and therefore client  $j$  would have turned grey and been connected to some facility  $i = \phi(j)$  that is temporarily open.

The high-level structure of the  $k$ -round algorithm is shown in Algorithm 5 and the above discussion is encapsulated in the following lemma.

LEMMA 8. *Every client that enters the outer for-loop of the  $k$ -round algorithm, colored white, turns grey during one of the  $k_1$  iterations of the loop and gets connected to some temporarily open facility.*

Also, similar to Lemma 4, we can show that the  $\alpha_j$ -values, when scaled down by a factor of  $n^{3/k_1}$  and the corresponding  $\beta_{ij}$  values form a feasible solution to the DP, yielding the following lemma; the proof is similar to the proof of Lemma 4 and is skipped.

LEMMA 9. *After the  $k$ -round Algorithm has terminated,  $\sum_{j \notin L} \alpha_j \leq n^{3/k_1} \cdot OPT$ .*

---

**Algorithm 5**  $k$ -round Algorithm

---

```
1: Initialization Phase.
2: for  $p \leftarrow 1$  to  $k_1$  do
3:   Run an iteration of the Primal-Dual Algorithm (for
     each client and each facility); the output is the set
     (denoted  $T_p$ ) of temporarily open facilities
4:   if  $(\exists \text{ facility } i \in T_p) \wedge$ 
      $(\exists \text{ facility } i' \in (T_1 \cup T_2 \cup \dots \cup T_{p-1})) \wedge$ 
      $(\exists \text{ client } j: \beta_{ij} > 0 \wedge \beta_{i'j} > 0)$  then
5:      $\text{status}(i) \leftarrow \text{closed}$  (delete  $i$  from  $T_p$ )
6:      $\phi(j) \leftarrow i'$ 
7:   end if
8:   for  $q \leftarrow 1$  to  $k_2$  do
9:     Run an iteration of the Sparsification Algorithm
10:    {for each facility that remains in  $T_p$ }
11:  end for
12: end for
```

---

## 3.1 The Sparsification Phase

### 3.1.1 Motivation

The Sparsification Phase is the fundamental challenge for designing a  $k$ -round algorithm. Recall from the analysis of the Logarithmic-round algorithm that the Sparsification Phase had two goals: the first goal is to bound, for each client  $j$ , the size of  $K_j = \{i \in F_t \mid \beta_{ij} > 0 \text{ and } i \text{ is open}\}$ . If we permanently open all the nodes in  $F_t$  (the set of temporarily open facilities),  $K_j$  can become arbitrarily large. This motivated the definition of the graph  $H = (F_t, E_t)$ , where  $E_t$  contains all edges connecting pairs of facilities  $i, i' \in F_t$  such that there is a client  $j$  with  $\beta_{ij} > 0$  and  $\beta_{i'j} > 0$ . Then an MIS  $M$  on  $H$  is computed and only nodes in  $M$  are permanently opened. As a result,  $|K_j| \leq 1$  for all  $j$ . The second goal of the Sparsification Phase is to guarantee that if for a client  $j$ , facility  $i = \phi(j)$  is shut down (due to not being in  $M$ ) then some neighbor  $i'$  of  $i$  in  $H$  is open and  $j$  can be reconnected to  $i'$  without the cost of the reconnection being too high relative to  $\alpha_j$ . In the next two subsections we explain in detail why it is critical to fulfill these two goals and how they are achieved.

Given that we now have a very small number of rounds in which to do sparsification, we can no longer use a Luby-like algorithm because it does not sparsify the graph rapidly enough to ensure adequate progress in a small number of rounds. Since the objective of the  $k$ -round algorithm is not to achieve a constant-factor approximation, we can relax the requirement that we need to shut down enough temporarily open facilities to ensure that  $|K_j| = O(1)$  for all clients  $j$ . Instead, we would like to use  $k_2$  rounds to sparsify  $H$  and select a set of nodes  $M$  that induces a sparse subgraph with maximum degree  $O(m^{2/k_2})$ , implying an upper bound of  $O(m^{2/k_2})$  on  $|K_j|$ .

Following a similar line of argument, unlike the Logarithmic round algorithm, we no longer need the reconnection cost of a client  $j$  to be bounded by a constant factor of  $\alpha_j$ . For this algorithm, it is enough to guarantee that if for a client  $j$ , facility  $i = \phi(j)$  is shut down (due to not being in  $M$ ) then some neighbor  $i'$  of  $i$  in  $H$  is open and  $j$  can be reconnected to  $i'$  without the cost of the reconnection exceeding  $O(m^{2/k_2} \cdot \alpha_j)$ .

In Subsection 3.1.2 we present an algorithm that runs for  $k_2$  rounds and selects a node subset  $M$  of  $H$  and guarantees,

with high probability, the following two properties:

1.  $\Delta(H[M]) \leq 6 \cdot m^{2/k_2}$ .
2.  $\text{distance}(i, M) \leq 2 \cdot k_2$  for all nodes  $i$  in  $H$ . Note that for all  $k_2 = O(\frac{\log m}{\log \log m})$ ,  $k_2 = O(m^{2/k_2})$ .

Here  $\Delta(\cdot)$  refers to the maximum degree of a given graph and  $\text{distance}(i, M)$  is the shortest number of hops in  $H$  between a facility  $i$  and the set of selected vertices  $M$ . Note that due to Property 2, a node in  $H$  may not find a node in  $M$  in its neighborhood, but will find, with high probability, some node  $i'$  in  $M$  within  $2 \cdot k_2$  hops. Now, if all the  $\alpha_j$ -values are identical, then we can guarantee that the cost of reconnecting  $j$  to  $i'$  is  $O(\alpha_j \cdot k_2)$ , as  $\alpha_j$  is an upper bound on every connection cost. This bound on the reconnection cost suffices for our analysis and in order to ensure that all of the  $\alpha_j$ -values are identical, we run the Sparsification Algorithm repeatedly and separately, on just the facilities that were temporarily opened in each Primal-Dual Algorithm iteration (Lines 7-10). This modification by itself would still allow  $K_j$  to be quite large because a client  $j$  could be making positive payments to facilities that were temporarily opened in different iterations. To ensure that  $|K_j|$  is bounded as desired, we also additionally shut down (in Lines 4-6) each facility  $i$  that was temporarily opened in an iteration  $p$  but which has a facility  $i'$  “nearby” that was temporarily opened in a previous iteration.

Our Sparsification Phase is inspired by a recent randomized MIS algorithm due to Gfeller and Vicari [7]. These authors are interested in quickly computing an MIS on growth-bounded graphs. While our graphs are not growth-bounded, the first phase of the Gfeller-Vicari algorithm runs on arbitrary graphs and here we essentially show that even a few iterations of a variant of this algorithm guarantees enough progress for our purposes.

### 3.1.2 The Sparsification Algorithm

We will describe the Sparsification Phase with respect to an arbitrary input graph  $G = (V, E)$ . It is worth noting that in the context of Algorithm 5, the Sparsification Phase is run on a graph  $H = (F_t, E_t)$  where  $F_t$  is the set of facilities temporarily opened in an iteration  $p$  of the Primal-Dual Phase (Line 3) and not immediately shut down in Line 5; as usual  $E_t$  is the set of edges connecting pairs of facilities  $i, i'$  in  $F_t$  for which there is a client  $j$  with  $\beta_{ij} > 0$  and  $\beta_{i'j} > 0$ .

Let  $M_0 := V$  and  $G_0 := G$ . For  $s \geq 1$ , let  $M_s$  be the set of nodes selected in iteration  $s$  of the Sparsification Phase when run on graph  $G_{s-1}$  and let  $G_s := G[M_s]$ . We now describe a typical iteration  $s$  of the Sparsification Phase. For ease of presentation, set  $r = s + 2$ . The goal of iteration  $s$  is to select a set  $M_s$  of nodes so that the maximum degree in  $G[M_s]$  is at most  $m^{2/r}$ . Let  $\tau := m^{2/r}$ . Any node in  $G_{s-1}$  with degree no greater than  $\tau$  is called a *small degree* node. Any small degree node, all of whose neighbors are also small degree nodes, is called an *interior* node; all other small degree nodes are called *boundary* nodes. Nodes that are not small degree nodes are called *exterior* nodes. See Figure 3 for an illustration of these definitions. The Sparsification Algorithm is shown in Algorithm 6.

### 3.1.3 Analysis

Since our goal is to ensure that all node degrees are bounded by  $m^{2/r}$ , the interior vertices are not of concern because of

---

**Algorithm 6** Sparsification Algorithm
 

---

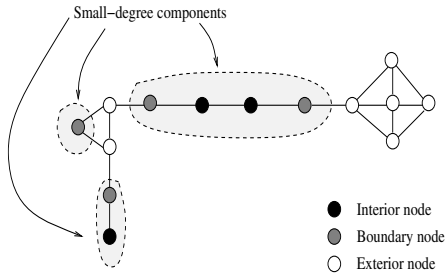
```

1: Input:  $G = (V, E)$ ,  $|V| = m$ , positive integer  $t$ 
2:  $M_0 \leftarrow V$ 
3: for  $s = 1$  to  $t$  do
4:    $r \leftarrow s + 2$ 
5:    $\tau \leftarrow m^{\frac{2}{r}}$ 
6:    $\rho \leftarrow \frac{1}{m^{\frac{1}{r}}}$ 
7:   for all  $u \in M_{s-1}$ ,  $d_u \leftarrow$  degree of  $u$  in  $G[M_{s-1}]$ 
8:   for all  $u \in M_{s-1}$ ,  $\Delta_u \leftarrow \max\{d_v \mid v \in N[u]\}$ 
     (here  $N[u]$  is the closed neighborhood of  $u$  in  $G[M_{s-1}]$ )
9:   if  $(d_u \leq \tau) \wedge (\Delta_u \leq \tau)$  then
10:     $type(u) \leftarrow interior$ 
11:     $u$  joins  $M_s$ 
12:   end if
13:   if  $(d_u \leq \tau) \wedge (\Delta_u > \tau)$  then
14:     $type(u) \leftarrow boundary$ 
15:   end if
16:   if  $(d_u > \tau)$  then
17:     $type(u) \leftarrow exterior$ 
18:   end if
19:   if  $type(u) == interior$  then
20:     $u$  joins  $M_s$  (deterministically)
21:   end if
22:   if  $type(u) \in \{boundary, exterior\}$  then
23:     $u$  independently joins  $M_s$  with probability  $\rho$ 
24:   end if
25: end for
26: Output:  $M_t$ 

```

---

their small degrees and their neighbors' small degrees and they can simply be included in the output. The boundary vertices and the exterior vertices probabilistically include themselves in the output. Since boundary vertices also have small degrees, we do not have to worry about their degrees. We can inductively upper bound the degree of exterior vertices entering iteration  $s$  and if we choose the probability  $\rho$  of inclusion in  $M_s$  carefully, then we can bound the degrees of exterior vertices by  $m^{2/r}$  also. This is proved in the next two lemmas. Of course, if the goal was solely to bound the degrees, then we could achieve this trivially by including no vertices in  $M_s$ . An additional and equally important goal in the context of Algorithm 5 is to show that after  $k_2$  iterations of the executed, the output  $M := M_{k_2}$  satisfies  $distance(i, M) \leq 2 \cdot k_2$  for all  $i \in V$ . Here  $distance(\cdot, \cdot)$  refers to hop distance in  $G$ . Lemma 13 shows that in each iteration the selected vertices can get at most 2 hops farther



**Figure 3:** Shows the partition of the nodes of a graph into interior, boundary, and exterior nodes with degree threshold  $\tau$  set to 2.

away from all vertices of  $G$ .

LEMMA 10. *For some constant  $c_1 > 0$ , for all  $s \leq c_1 \cdot \frac{\log m}{\log \log m}$ , if the maximum degree in  $G_{s-1}$  is at most  $m^{\frac{3}{(s+2)}}$ , then with high probability, the maximum degree of  $G_s$  is at most  $6 \cdot m^{\frac{2}{(s+2)}}$ .*

PROOF. Recall that we use  $r$  to denote  $s + 2$ . For a node  $u$  in  $G_s$ , if  $u$  is either an interior or a boundary node during iteration  $s$ , then the degree of  $u$  in  $G_{s-1}$  is at most  $m^{2/r}$ . As vertices are only dropped during the iteration, the degree of  $u$  in  $G_s$  is also at most  $m^{2/r}$ . Hence we can simply consider the case when  $u$  is an exterior node in iteration  $s$ . Let  $X_u$  be the random variable denoting the number of neighbors of  $u$  in  $G_{s-1}$  who joined  $M_s$ . Since  $u$  has at most  $m^{\frac{2}{r}}$  neighbors in  $G_{s-1}$  and each of them can join  $M_s$  with probability  $\frac{1}{m^{\frac{1}{r}}}$ ,

$$E[X_u] \leq m^{\frac{2}{r}} \cdot \frac{1}{m^{\frac{1}{r}}} = m^{\frac{1}{r}}$$

Since  $X_u$  is the sum of independent binary random variables, using a simple version of Chernoff bounds, we get that

$$Prob[X_u \geq 6m^{\frac{2}{r}}] \leq 2^{-6m^{\frac{2}{r}}}$$

Since  $r \leq c_1 \cdot \frac{\log m}{\log \log m} - 2$ ,

$$Prob[X \geq 6m^{\frac{2}{r}}] \leq 2^{-6m^{\frac{2}{r}}} \leq 2^{-6m^{\frac{\log \log m}{c_1 \log m}}}$$

It is easy to verify that for some constant  $c_1 > 0$ , the right hand side above is bounded by  $1/m^2$ . This proves that with probability at least  $1 - 1/m^2$ , the maximum degree of node  $u$  in  $G_s$  is at most  $6m^{\frac{2}{r}}$ . Using the union bound, we see that the probability that all vertices (at most  $m$ ) have degree at most  $6m^{\frac{2}{r}}$  in  $G_s$  is at least  $(1 - m \cdot \frac{1}{m^2}) = (1 - \frac{1}{m})$ .  $\square$

The hypothesis of the above lemma required that the maximum degree of  $G_{s-1}$  be upperbounded by  $m^{\frac{3}{s+2}}$ . We now inductively show that this is true.

LEMMA 11. *For some constant  $c_2 > 0$ , for all  $s \leq c_2 \cdot \log m$ , the maximum degree in  $G_{s-1}$  is at most  $m^{\frac{3}{s+2}}$ , with high probability.*

PROOF. We show this by induction. The base case of  $s = 1$  is trivial since every node has degree at most  $m$  at all times. For the inductive step, consider iteration  $s > 1$ . Assuming, using the inductive hypothesis, that the maximum degree in  $G_{s-1}$  is  $m^{\frac{3}{(s+2)}}$ , we get by Lemma 10, that with high probability the maximum degree in  $G_s$  is at most  $6m^{\frac{2}{(s+2)}}$ . It is easy to verify that for large enough  $m$ , there exists a constant  $c_2 > 0$  such that for all  $s \leq c_2 \cdot \log m$ ,  $6m^{\frac{2}{(s+2)}} \leq m^{\frac{3}{(s+3)}}$ .  $\square$

Lemma 11 proves the pre-condition assumed in Lemma 10. The following corollary follows directly from these two lemmas.

COROLLARY 12. *For any  $k_2 \leq c_1 \cdot \frac{\log m}{\log \log m}$  for some constant  $c_1 > 0$ , after  $k_2 - 2$  rounds, the Sparsification Algorithm selects a node subset  $M$  of  $H$  and guarantees, with high probability, that  $\Delta(H[M]) \leq 6 \cdot m^{2/k_2}$ .*

The next lemma shows that with each iteration of the Sparsification algorithm, the selected set of nodes “move” at most 2 hops further away from the rest of the graph.

LEMMA 13. *For some constant  $c_3$  and for  $s \leq c_3 \cdot \frac{\log m}{\log \log m}$ , with probability at least  $1 - 1/m^2$ , for every node  $u$  in  $M_{s-1}$ , there is a node  $v$  in  $M_s$  such that  $v$  is in the 2-neighborhood of  $u$  in  $M_s$ .*

PROOF. Consider a node  $u$  in  $M_{s-1}$ . If  $u$  is an interior node in iteration  $s$ , it is included in  $M_s$ . If  $u$  is an exterior node in iteration  $s$ , we know that the degree of  $u$  in  $G_{s-1}$  is greater than  $m^{2/r}$ . The probability that none of  $u$ 's neighbors are selected for  $M_s$  is at most

$$\left(1 - \frac{1}{m^{1/r}}\right)^{m^{2/r}} \leq e^{-m^{3/r}}.$$

It is easy to verify that there exists a constant  $c_3 > 0$ , such that for all  $r \leq c_3 \cdot \frac{\log m}{\log \log m} - 2$ , this probability is at most  $1/m^2$ . If  $u$  is a boundary node, then it has at least one exterior node as a neighbor and therefore, using what we have shown for exterior nodes, we conclude that with probability at least  $1 - 1/m^2$ , there is node from  $M_s$  in  $u$ 's 2-neighborhood.  $\square$

### 3.2 Wrapping Things Up

Recall that the Sparsification Phase is applied to a graph  $H = (F_t, E_t)$ , where  $F_t$  is the set of facilities temporarily opened in an iteration  $p$  of the Primal-Dual Phase (Line 3) and not immediately shut down in Line 5 and  $E_t$  is the set of edges connecting pairs of facilities  $i, i'$  in  $F_t$  for which there is a client  $j$  with  $\beta_{ij} > 0$  and  $\beta_{i'j} > 0$ . The following lemma is obtained in a straightforward way by repeatedly applying Lemma 13.

LEMMA 14. *There is a constant  $c > 0$  such that for any  $k_2 \leq c \cdot \frac{\log m}{\log \log m}$ , after  $k_2$  iterations of the Sparsification Algorithm on input  $H$ , the output  $M := M_{k_2}$  satisfies, with high probability, the property that  $\text{distance}(i, M) \leq 2k_2$  for all nodes  $i$  in  $H$ . Here,  $\text{distance}(i, M)$  is the shortest path distance in hops in  $H$  from node  $i$  to some node in  $M$ .*

PROOF. Let  $T_i$  be the nodes selected in the  $i$ th iteration and let us set  $T_0 = V$ . Due to Lemma 13, with high probability, there exists a vertex in  $T_i$  which is at most 2 hops away from every vertex in  $T_{i-1}$  for  $i > 0$ . Hence the claim follows by induction over  $i$ .  $\square$

The next lemma shows that our efforts at careful sparsification have paid off in terms of guaranteeing that when a client  $j$  is reconnected, its reconnection cost can still be charged to  $\alpha_j$ .

LEMMA 15. *At the end of the  $k$ -round algorithm, each client  $j$ , with high probability, will connect to open facility  $i'$ , with  $c_{i'j} \leq (4k_2 + 1) \cdot \alpha_j$ .*

PROOF. If a client  $j$  is directly connected to a facility  $i'$ , then we simply have  $c_{i'j} \leq \alpha_j$ . If client  $j$  is indirectly connected to facility  $i'$ , there could be two possible reasons for it:

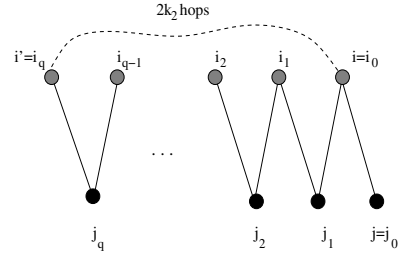


Figure 4: Client  $j$  was initially connected to facility  $i'$  that was closed during sparsification. In that case,  $j$  will find a facility  $i$  to connect to so that  $\text{distance}(i, i') \leq 2k_2$ .

- (i)  $j$  was directly connected to a facility  $i$  that was shut down in Line 5 of the  $k$ -round algorithm (Algorithm 5). Say this happened in the  $p$ -th iteration of the outer loop. Hence  $i$  just became temporarily open in this iteration (i.e., the  $p$ -th iteration). From the conditions stated in Line 4 of Algorithm 5, there must also be another facility  $i'$  that became temporarily open at an earlier iteration (say  $p'$ ,  $p' < p$ ) and a client  $j'$  such that  $\beta_{i'j'} > 0$  and  $\beta_{ij'} > 0$ . Hence,  $c_{i'j'} \leq \alpha_{j'}$  and  $c_{ij'} \leq \alpha_{j'}$ . Notice that  $\alpha_{j'}$  cannot increase after the end iteration  $p'$  as a facility that client  $j'$  has positively contributed to would be temporarily open. Hence,  $\alpha_{j'} < \alpha_j$ . Also,  $j$  was directly connected to  $i$ , hence  $c_{ij} \leq \alpha_j$ . Using the triangle inequality,

$$c_{i'j} \leq c_{ij} + c_{ij'} + c_{i'j'} \leq \alpha_j + \alpha_{j'} + \alpha_{j'} < 3\alpha_j$$

- (ii)  $j$  was directly connected to a facility  $i$  that was shut down during an iteration of the Sparsification Algorithm (Line 8) of the  $k$ -round algorithm (Algorithm 5). From Lemma 14, it follows that there is a temporarily open facility  $i'$ , such that  $\text{distance}(i', i) \leq 2k_2$  (see Figure 4). For each pair  $i_q$  and  $i_{q+1}$ , to be neighbors in  $H$ , they must have had a client  $j_{q+1}$  such that  $c_{i_q j_{q+1}} \leq \alpha_{j_{q+1}}$  and  $c_{i_{q+1} j_{q+1}} \leq \alpha_{j_{q+1}}$ . Also  $i_q$  and  $i_{q+1}$  have been temporarily opened, which means  $\alpha_{j_{q+1}}$  cannot grow beyond this iteration, i.e.  $\alpha_{j_{q+1}} \leq \alpha_j$ . Given,  $j$  was directly connected to  $i$ , hence  $c_{ij} \leq \alpha_j$ . Again, using the triangle inequality we get  $c_{i'j} \leq (4k_2 + 1) \cdot \alpha_j$ .

$\square$

Our final theorem shows the approximation guarantee for the solution produced by the  $k$ -round algorithm. The proof is similar to the proof of Lemma 7 and is skipped.

**Theorem 16.** *For any  $k = k_1 \cdot k_2$ , with  $k_2 = O\left(\frac{\log m}{\log \log m}\right)$ , the  $k$ -round algorithm outputs an  $O\left(n^{\frac{3}{k_1}} m^{\frac{2}{k_2}}\right)$ -approximation to the metric facility location problem.*

## 4. CONCLUSIONS

The main contribution of this paper is the demonstration of a trade-off between the approximation factor and the number of rounds of communication for the metric facility location problem in the *CONGEST* model. Specifically,

we present an algorithm that runs in  $k$  rounds and yields an  $O(m^{\frac{2}{\sqrt{k}}} \cdot n^{\frac{3}{\sqrt{k}}})$ -approximation. Setting  $k = c \cdot \log^2(m + n)$  for some constant  $c$ , would make this approximation factor reduce to a constant. However, our randomized, rapid Sparsification Algorithm does not exhibit the desired behavior when  $k > c' \cdot \left(\frac{\log m}{\log \log m}\right)^2$ , for some constant  $c'$ . So there is a small gap at the upper end in the desired range of values that  $k$  can take and plugging this gap is a problem that we intend to tackle. Without closing this gap, we cannot claim that the approximation factor we get for *metric* facility location is any better than the approximation factor obtained by Moscibroda and Wattenhofer [17] for *non-metric* facility location. Assuming that  $m = n$ , we see that as  $k$  exceeds  $\Theta\left(\left(\frac{\log m}{\log \log m}\right)^2\right)$ , the quantity  $O(m^{\frac{2}{\sqrt{k}}} \cdot n^{\frac{3}{\sqrt{k}}}) = O(m^{\frac{5}{\sqrt{k}}})$  becomes sublogarithmic, reaching a constant when  $k$  reaches  $\Theta(\log^2 m)$ . For smaller values of  $k$ , our approximation factor is incomparable with the approximation factor obtained by Moscibroda and Wattenhofer [17] and which is better depends on the value of  $\rho$  that appears in their approximation factor.

Gehweiler et al. [6] obtain a  $O(1)$  approximation for the metric facility location problem with *uniform* facility opening costs. The Logarithmic-round Algorithm in this paper solves the non-uniform version of Gehweiler's problem in  $O(\log n)$  rounds. Are  $O(\log n)$  rounds necessary? Is it possible to obtain non-trivial lower bounds in this setting? Moscibroda and Wattenhofer [17] call the problem of finding lower bounds in this model an "outstanding" open problem.

Another question that interests us is whether it is possible to obtain a better approximation factor versus communication rounds trade-off for the *non-metric* facility location, improving on the approximation factor achieved by Moscibroda and Wattenhofer [17] for  $k$ -round algorithms.

## 5. REFERENCES

- [1] Noga Alon, László Babai, and Alon Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7(4):567–583, 1986.
- [2] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and  $k$ -median problems. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 378, Washington, DC, USA, 1999. IEEE Computer Society.
- [3] G. Cornuejols, G. Nemhouser, and L. Wolsey. *Discrete Location Theory*. Wiley, 1990.
- [4] Michael Elkin. Distributed approximation: a survey. *SIGACT News*, 35(4):40–57, 2004.
- [5] Christian Frank. *Algorithms for Sensor and Ad Hoc Networks*. Springer, 2007.
- [6] Joachim Gehweiler, Christiane Lammersen, and Christian Sohler. A distributed  $O(1)$ -approximation algorithm for the uniform facility location problem. In *SPAA '06: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 237–243, 2006.
- [7] Beat Gfeller and Elias Vicari. A randomized distributed algorithm for the maximal independent set problem in growth-bounded graphs. In *PODC '07: Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 53–60, 2007.
- [8] Dorit S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22(1):148–162, 1982.
- [9] Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and  $k$ -median problems using the primal-dual schema and Lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.
- [10] M. Khan and G. Pandurangan. A fast distributed approximation algorithm for minimum spanning trees. *Distributed Computing*, 20(6):391–402, 2008.
- [11] Fabian Kuhn and Thomas Moscibroda. Distributed approximation of capacitated dominating sets. In *SPAA '07: Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, pages 161–170, New York, NY, USA, 2007. ACM.
- [12] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 300–309, New York, NY, USA, 2004. ACM.
- [13] Fabian Kuhn and Roger Wattenhofer. Constant-time distributed dominating set approximation. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 25–32, New York, NY, USA, 2003. ACM.
- [14] Christoph Lenzen, Yvonne Anne Oswald, and Roger Wattenhofer. What can be approximated locally?: case study: dominating sets in planar graphs. In *SPAA '08: Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures*, pages 46–54, New York, NY, USA, 2008. ACM.
- [15] Jyh-Han Lin and Jeffrey Scott Vitter.  $\epsilon$ -approximations with minimum packing constraint violation (extended abstract). In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 771–782, 1992.
- [16] M Luby. A simple parallel algorithm for the maximal independent set problem. In *STOC '85: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 1–10, 1985.
- [17] Thomas Moscibroda and Roger Wattenhofer. Facility location: distributed approximation. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 108–117, 2005.
- [18] Saurav Pandit and Sriram Pemmaraju. Finding facilities fast. In *Proceedings of the 10th International Conference on Distributed Computing and Networks*, January 2009.
- [19] David Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, 2000.
- [20] David B. Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems (extended abstract). In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 265–274, 1997.