2. Facility Location

Christian Frank

2.1 Introduction

The facility location problem has occupied a central place in operations research since the early 1960's, as it models design decisions on the placement of factories, warehouses, schools, or hospitals to serve a set of customers efficiently. For an overview of previous work see the survey chapter by Cornuejols, Nemhauser & Wolsey in [21]. Recently, it has received renewed research interest because it can be used to place servers or proxies in communication networks [28].

We apply the facility location problem to configuration and deployment issues in wireless sensor networks. More specifically, we discuss how algorithms for the facility location problem can be used to assign functionality to certain nodes in a wireless sensor network, i.e., choose a subset of designated nodes as *service providers* of a particular service.

Such assignment of functionality to designated nodes is a key element in the often-cited ability for *self-configuration* [3] that is required in many sensor-network applications. Consider clustering, which is a prominent sensor network configuration problem (see Chapter **??**), where a subset of cluster leaders is selected to provide a service to their neighboring nodes, while the latter (slave nodes) can shut down some functionality and thus save power. Clustering can for instance be used to improve the efficiency of data delivery (e.g., flooding, routing). In this setting, cluster leaders act as sole direct communication partners for the associated slave nodes, which allows slave nodes to synchronize to their cluster leader's transmission schedule and shut down their transceivers based on this schedule.

As slave nodes save power, it is desirable to have only few cluster leaders (as long as each slave node has a cluster leader as a neighbor) to enable energy savings at many slave nodes. In this sense an optimal configuration is modeled by the minimum dominating set problem discussed in Chapter **??**. An example of such a configuration is shown in Figure 2.1(a).

However, such a strategy increases the distance between cluster leaders and their associated nodes. If network nodes can vary their transmission power, such a configuration is not energy optimal. In particular, an energy-optimal configuration would require placing cluster leaders as close as possible to their associated nodes. An example of this strategy is shown in Fig-

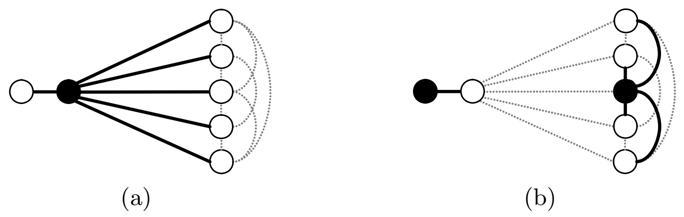


Fig. 2.1. Different clustering configurations

ure 2.1(b). Here an optimal balance between transmission energy expenditure and the effort involved in operating cluster leaders must be found.

Apart from clustering, solving the facility location problem is useful for deploying services to certain nodes in the network. This will be increasingly relevant as wireless sensor networks pursue the path of other distributed systems platforms towards re-usable component-based systems that are supported by adequate deplyoment middleware (e.g., [6, 11]). While present research prototypes (see Chapter **??**) are still based on *application-specific* design and implementation of major parts of the system, a key for the wide-spread adoption of sensor-networks is to provide a large pool of re-usable and pluggable components that can be used to build a new application. Such components could provide a service for their network neighborhood and thus need not be executed on *every* single node. *Where* such components will be executed in the network will be addressed by distributed algorithms solving the facility location problem.

In the following, we formally introduce the facility location problem in Section 2.2, discuss centralized approximations of the problem in Section 2.3, and describe distributed approximation algorithms in detail in Sections 2.4 and 2.5. We will conclude the chapter with an outlook on how the presented approaches can provide a basis for designing *local* facility location algorithms that are particularly suited for wireless sensor networks in Section 2.6.

2.2 Problem Definition

The facility location problem formulates the geographical placement of *facilities* to serve a set of *clients* or customers, sometimes also called cities. We will define the traditional facility location problem in Section 2.2.1. In wireless sensor networks *facilities* can represent any kind of service provider (like cluster heads, time synchronization hubs, or network sinks) that can be used by *client* nodes nearby. We will discuss in more detail how the facility location problem can be applied to the multi-hop wireless sensor network model in the subsequent Section 2.2.2.

2.2.1 Problem Formulations

A number of problem formulations exist in the domain of the facility location problem. Generally, these are based on connecting clients to a set of open facilities at minimum cost.

Definition 2.2.1 (Uncapacitated Facility Location Problem – UFL). *Given a bipartite graph G with bipartition (C, F) where C is a set of clients and F is a set of potential facilities, a fixed cost $f_i \in \mathbb{R}_+$ for opening each facility $i \in F$ and connecting costs $c_{ij} \in \mathbb{R}_+$ for connecting client $j \in C$ to facility $i \in F$: Return a subset $S \subseteq F$ of* open *facilities and a mapping $\sigma : C \to S$ of clients to open facilities that minimize the total costs (fixed costs of open facilities and connection costs),*

$$\sum_{i \in S} f_i + \sum_{j \in C} c_{\sigma(j)j} \ .$$

Figure 2.2 sketches the elements introduced above. We will use $n = |C|$ and $m = |F|$ to refer to the number of clients and facilities, respectively.

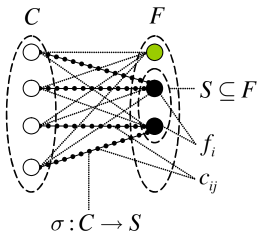


Fig. 2.2. Elements of the facility location problem

In the above UFL formulation, the fixed costs f_i enable the trade-off to open additional facilities at additional costs if these can reduce connecting costs. Alternatively, in the following problem formulation, opening costs f_i are zero. Instead, a fixed number of k facilities will be chosen from F such that the sum of the distances from each client to its next facility is minimized. This models the problem of finding a minimum-cost clustering of k clusters [27, chap. 25].

Definition 2.2.2 (The k-Median Problem – k-MN). *Given a bipartite graph G with bipartition (C, F) where C is a set of clients and F is a set of potential facilities, an integer k limiting the number of facilities that can be opened, and connecting costs $c_{ij} \in \mathbb{R}^+$ for each $i \in F$ and $j \in C$: Return a subset $S \subseteq F$ of $|S| \leq k$* open *facilities and a mapping $\sigma : C \to S$ of clients to open facilities that minimize the total connection costs,*

$$\sum_{j \in C} c_{\sigma(j)j} \ .$$

Notice that the facility location problem is a rather generic problem formulation. For example, it is a generalized version of the minimum dominating set (MDS) problem. Consider a graph $G = (V, E)$ which is an instance of MDS. It can be reduced to UFL if F and C are set to be a copy of the same vertices V, $c_{ij} = 0$ for $(i, j) \in E$ and ∞ otherwise, and the opening costs f_i are set to a constant f_c for all i.

Due to its wide range of applications, many variants of the facility location problem exist. A common generalization of UFL and k-MN is $k-$UFL that includes non-zero opening costs and the additional constraint that at most k facilities can be opened. The k-UFL problem corresponds to k-MN if opening costs f are zero and to UFL if k is set to $|F|$. The *capacitated* facility location problem refers to an additionally constrained version of UFL, in which each potential facility $i \in F$ can serve at most u_i clients. Last but not least, an instance of UFL or k-MN is called *metric* if the connection costs c_{ij} satisfy the triangle inequality, which requires that, given any three nodes i, j, k the cost of the direct edge (i, j) is smaller than the path over an additional hop $(c_{ij} \leq c_{ik} + c_{kj})$. Note that this implies that the edge (i, j) must exist if (i, k) and (k, j) exist and therefore a metric instance must be complete or at least consist only of cliques.

2.2.2 Sensor Network Models

The above “traditional” definition of UFL and k-MN problems does not immediately accommodate the multi-hop network model used in wireless sensor networks. In multi-hop sensor networks, the network communication graph $G = (V, E)$ will be “unfolded” into a UFL instance in various ways, depending on the given application.

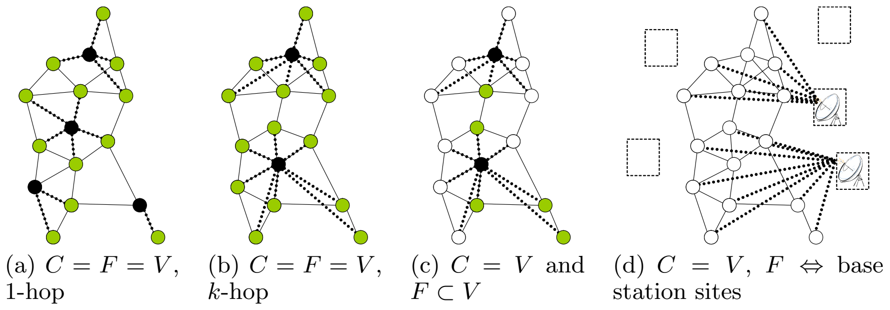

(a) $C = F = V$, 1-hop (b) $C = F = V$, k-hop (c) $C = V$ and $F \subset V$ (d) $C = V$, $F \Leftrightarrow$ base station sites

Fig. 2.3. Possible applications of the facility location problem in wireless sensor networks

In our introductory examples of finding an energy-efficient clustering or deploying services onto network nodes, we have regarded special cases of the facility location problem, in which all nodes V of the network communication graph simultaneously take on the role of a client and of a potential facility, thus $F = V$ and $C = V$. Specifically, for an energy efficient clustering it is required that clients connect to facilities which are their direct neighbors (Figure 2.3(a)). When deploying services to network nodes, it is sometimes sufficient for clients to connect to service provider nodes that are multiple network hops away (Figure 2.3(b)), which allows for a smaller number of service nodes.

In other settings, only some nodes have the necessary capabilities (e.g., remaining energy, available sensors, communication bandwidth, or processing power) to execute a service. In these cases, a subset of eligible nodes can be selected as facilities based on their capabilities beforehand [8], which results in $C = V$ but $F \subset C$ (Figure 2.3(c)).

Another application of UFL is to obtain an optimal deployment of base stations (sometimes referred to as *sinks* or *gateways*) to connect the sensor network to a background infrastructure. Here F contains a set of potential basestation sites, while installed base stations then provide back-end connectivity to all network nodes ($C = V$). In this case, S represents the minimum-cost deployment of an optimally placed number of base stations (Figure 2.3(d)).

In all cases, c_{ij} denote some kind of communication cost between a facility node i and a client node j. In a communication graph $G = (V, E)$ of a multi-hop network, we can assume that edge costs are given for all $(i, j) \in E$, which represent the quality of the link, e.g., based on dissipated energy, latency, or interference. Depending on the indicator used to represent link quality, the resulting facility location instance may not be *metric*. However, we can obtain a metric instance if one sets c_{ij} to the costs of the shortest i-j-path in G, which additionally allows to address scenarios in which clients and facilities are allowed to be an arbitrary number of network hops apart.

In the remaining chapter, algorithms will be presented in terms of the standard bipartite graph model as given in Definition 2.2.1. This model can directly be applied to placing base stations (Figure 2.3(d)), and may further be used, after duplicating each network node into a facility and client node interconnected with zero costs, to compute an energy efficient clustering (Figure 2.3(a)). However, when facilities can be multiple hops away (Figures 2.3(b) and 2.3(c)), additional adaptations of the distributed algorithms we present in this chapter are required. We will preview such adaptations in Section 2.6.

2.3 Centralized Approximations

In this section, we discuss centralized approximation schemes for UFL problems. Before doing so, we present a lower bound for approximating the non-metric version of UFL, resulting from a reduction of the set cover problem.

2.3.1 Negative Results and Set Cover

It turns out that an instance of the well-known *set cover problem* can be directly transformed into a *non-metric* instance of UFL [28]. The *set cover problem* is defined as follows:

Definition 2.3.1 (Set Cover Problem). *Given a pair $(U, \mathbf{S})$, where U is a finite set and $\mathbf{S}$ is a family of subsets of U with $\bigcup_{S \in \mathbf{S}} S = U$, and weights $c : \mathbf{S} \to \mathbb{R}^+$ for each such subset: Find a set $\mathcal{R} \subseteq \mathbf{S}$ with $\bigcup_{S \in \mathcal{R}} S = U$ that minimizes the total weight $\sum_{S \in \mathcal{R}} c(S)$.*

There are well-known lower bounds for the approximation of the *set cover problem*. Raz and Safra [24] showed that there exists a constant $\mathcal{X} > 0$ such that, unless P=NP, there is no polynomial-time approximation algorithm that produces, in the worst-case, a solution whose cost is at most $\mathcal{X} \ln |U|$ times the optimum. Feige [7] showed that such an algorithm does not even exist for any $\mathcal{X} < 1$ unless every problem in NP can be solved in $O(n^{O(\log \log n)})$ time.

Reducing the *set cover problem* to UFL is straightforward: Let $(U, \mathbf{S}, c)$ be an instance of the *set cover problem*. We construct an instance of UFL, in which $C := U$, $F := \mathbf{S}$, and opening costs $f_s := c(S)$ for each $S \subseteq \mathbf{S}$. Further, for each $j \in C$ and $S \in F$, we set connection costs c_{Sj} to zero if $j \in S \in \mathbf{S}$ and to ∞ if $j \in U \setminus S$.

As a consequence, the above negative results directly apply to the *non-metric* version UFL and the best-possible approximation factor of a polynomial algorithm is logarithmic in the number of clients.

2.3.2 A Simple Greedy Algorithm

Conversely, Chvátal's algorithm [5] already provides a $1 + \ln |U|$ approximation of the *set cover* problem, and thus, a reduction of UFL to *set cover* will already provide a good approximation for UFL. For this reduction, we need the notion of a *star* (i, B) consisting of a facility i and an arbitrary choice of clients $B \subseteq C$. The *cost* of a star (i, B) are $f_i + \sum_{j \in B} c_{ij}$ while its *cost efficiency* is defined as the star's *cost* per connected client in B:

$$c(i, B) = \frac{f_i + \sum_{j \in B} c_{ij}}{|B|} \tag{2.1}$$

Given an instance (C, F, c, f) of UFL, we can generate a *set cover* instance by setting $U := C$ and $\mathbf{S} := 2^C$, where $c(B)$ of $B \in 2^C$ are set to the minimum cost of a star (i, B) among the possible stars built using different facilities i, namely to $c(B) = \min_{i \in F} f_i + \sum_{j \in B} c_{ij}$.

Because the resulting *set cover* instance has exponential size, it cannot be generated explicitly. Nonetheless, one can apply Chvátal's algorithm [5] by considering only the most relevant star in each step as proposed by Hochbaum [12]. This method (Algorithm 1) provides the first algorithm approximating the *non-metric* version of UFL.

Algorithm 1: Chvátal's algorithm applied to UFL

```
set $U = C$
while $U \neq \emptyset$ do
    find most cost-efficient star $(i, B)$ with $B \subseteq U$
    open facility $i$ (if not already open)
    set $\sigma(j) = i$ for all $j \in B$
    set $U = U \setminus B$
```

The algorithm greedily finds the most cost-efficient star (i, B), opens the facility i, connects all clients $j \in B$ to i, and from this point on disregards all (now connected) clients in B. The algorithm terminates when all clients are connected. Note that in spite of there being exponentially many sets $B \subseteq U$, the most efficient star can be found in polynomial time: For each facility i, clients j can be stored pre-sorted by ascending connection costs to i. Given some number k, the most cost-efficient star with k clients will consist of a facility i and the first k clients that have lowest connection costs to i – all other subsets of k clients can be disregarded, as these cannot be more efficient. Hence, at most n different sets must be considered (instead of 2^n).

The results for *set cover* [5] therefore imply that Algorithm 1 provides an approximation ratio of $1 + \ln |C|$ for the non-metric version of UFL, which is very close to its lower bound. For details on Chvátal's set cover algorithm [5] see also Theorem 16.3 in [15].

2.3.3 Improvements for Metric UFL

Jain et al. [13] showed that in the metric case, Algorithm 1 does not perform much better, namely its approximation is still at least $\log n / \log \log n$. However, a slight revision [13], which is shown in Algorithm 2, can guarantee a much better approximation for the metric case of UFL.

Interestingly enough, this algorithm – discovered and analyzed 21 years later – only adds one line to the end of the loop: Once a facility i has been opened, its opening cost f_i is set to zero. This allows facility i to be chosen again to connect additional clients in later iterations, based on a cost-efficiency that disregards i's opening costs f_i – as the facility i has already

Algorithm 2: Centralized *metric* UFL algorithm by Jain et al.

```
set $U = C$
while $U \neq \emptyset$ do
    find most cost-efficient star $(i, B)$ with $B \subseteq U$
    open facility $i$ (if not already open)
    set $\sigma(j) = i$ for all $j \in B$
    set $U = U \setminus B$
    set $\mathbf{f_i = 0}$
```

been opened before in order to serve other clients. This slight change – together with an analysis we will outline in the next section – provides a constant 1.861-approximation of the *metric* UFL problem. Note that this algorithm can be executed in a distributed manner, albeit in a linear number of rounds. See Section 2.4 for details.

2.3.4 LP-Duality based Analysis

Algorithm 2 provides a strikingly improved approximation guarantee for the metric case. Its analysis is based on interpreting its basic steps as updates of variables in a primal-dual pair of linear programs that formulate the facility location problem.

Specifically, UFL is formulated in terms of the integer linear program (ILP) shown below. Based on this formulation [28], various approximation schemes have been developed – including the distributed approximations we will present in this chapter. The ILP contains two different sets of binary variables y_i and x_{ij}, where y_i indicate that facility i is opened and x_{ij} that client j is connected to facility i.

$$\begin{array}{rrcll}
\text{minimize} & \sum_{i \in F} f_i y_i & + & \sum_{i \in F}\sum_{j \in C} c_{ij} x_{ij} & \\
\text{subject to} & \sum_{i \in F} x_{ij} & \geq & 1 & \forall j \in C \quad (1) \\
& x_{ij} & \leq & y_i & \forall i \in F, \forall j \in C \quad (2) \\
& x_{ij}, y_i & \in & \{0,1\} & \forall i \in F, \forall j \in C \quad (3)
\end{array} \qquad (2.2)$$

Constraint (1) ensures that each client is connected to at least one facility (the cost function and positive c_{ij} ensure that no more than one facility is connected) while (2) demands that a client can only be connected to a facility i if the facility is open. The corresponding LP relaxation of UFL (we will hence call it UFL$_{frac}$) changes the integer constraints (3) to $x_{ij}, y_i \geq 0$.

The dual linear program of UFL_{frac} is

$$\begin{array}{llrll} \text{maximize} & \sum_{j \in C} \alpha_j & & & \\ \text{subject to} & \sum_{j \in C} \beta_{ij} & \leq & f_i & \forall i \in F \\ & \alpha_j - \beta_{ij} & \leq & c_{ij} & \forall i \in F, \forall j \in C \\ & \beta_{ij}, \alpha_j & \geq & 0 & \forall i \in F, \forall j \in C \quad . \end{array} \tag{2.3}$$

In LP-duality based schemes, each iteration of a combinatorial algorithm (e.g., Algorithm 2 above) is interpreted as an update of the primal variables y_i and x_{ij}, and of the corresponding dual variables α_j and β_{ij}. The primal updates simply reflect the corresponding steps taken in the algorithm, e.g., if the algorithm opens facility i, the respective y_i is set to 1, while the dual updates are used to provide a certain approximation guarantee.

Recall the weak duality theorem: If $\mathbf{p} = (x_{ij}, y_i)$ and $\mathbf{d} = (\alpha_j, \beta_{ij})$ are feasible solutions for the primal and dual program, respectively, then the dual objective function bounds the primal objective function (for an introduction to LP-duality see [27, chap. 12]). That is, for a minimization problem like UFL, the costs of the optimal primal solution $\mathbf{p}_{\text{opt}}$ lie between

$$g(\mathbf{d}) \leq f(\mathbf{p}_{\text{opt}}) \leq f(\mathbf{p}) \tag{2.4}$$

where $f(\mathbf{p})$ and $g(\mathbf{d})$ denote the primal and the dual objective functions, respectively. Moreover, $g(\mathbf{d}) = f(\mathbf{p}_{\text{opt}}) = f(\mathbf{p})$, iff all of the so-called *complementary slackness conditions* are satisfied. For the programs in (2.2) and (2.3), the complementary slackness conditions are

$$\forall i \in F, j \in C: \quad x_{ij} > 0 \Rightarrow \quad \alpha_j - \beta_{ij} = c_{ij} \tag{2.5}$$

$$\forall i \in F: \quad y_i > 0 \Rightarrow \quad \sum_{j \in C} \beta_{ij} = f_i \tag{2.6}$$

$$\forall j \in C: \quad \alpha_j > 0 \Rightarrow \quad \sum_{i \in F} x_{ij} = 1 \tag{2.7}$$

$$\forall i \in F, j \in C: \quad \beta_{ij} > 0 \Rightarrow \quad y_i = x_{ij}. \tag{2.8}$$

These laws of LP-duality allow designing approximation algorithms in two specific ways. The first class of algorithms is based on the so-called *primal-dual* scheme: The algorithm produces a pair of *feasible* solutions $\mathbf{p}$ and $\mathbf{d}$ and in its iterations tries to satisfy more and more of the conditions (2.5-2.8). In particular, the approximation algorithm [14], which we will detail in the next section, satisfies all conditions but condition (2.5). Instead, it satisfies a relaxed version of the condition, with its implication relaxed to $(1/3)c_{ij} \leq \alpha_j - \beta ij \leq c_{ij}$. As all other conditions are satisfied, it can be shown that for each result of the algorithm consisting of a primal solution $\mathbf{p}$

and a dual solution $\mathbf{d}$, it holds that $g(\mathbf{d}) \leq f(\mathbf{p}) \leq 3 \times g(\mathbf{d})$. This implies that a primal solution $\mathbf{p}$ is provided with an approximation guarantee of 3 for the metric version of the facility location problem. We describe this algorithm in detail in the next Section 2.3.5.

A second class of algorithms, instead, employs the scheme of *dual fitting*, which provides a basis for both the analysis of Algorithm 2 and the analysis of a fast distributed algorithm we will present in Section 2.5. *Dual fitting* based algorithms perform the updates of the primal and dual variables such that these obey *all* complementary slackness conditions, i.e., the two objective functions are equal throughout the algorithm (in the general case, the primal is at least bounded by the dual).

However, while the primal variables represent a feasible solution of the primal program, the dual variables are allowed to become *infeasible*. Nevertheless, the authors show that the degree of infeasibility of α_j and β_{ij} is bounded and that – after dividing all dual variables by a suitable factor γ – the dual is feasible and does *fit* into the dual LP, that is, satisfies all dual constraints. In [13], the authors then formulate an additional LP (the so-called *factor-revealing* LP) to find the minimal γ that suffices for all problem instances. For Algorithm 2 they give $\gamma = 1.861$. By the laws of LP duality (2.4) and considering the objective functions for UFL, this implies that Algorithm 2 provides a 1.861-approximation for metric UFL instances.

2.3.5 A 3-Approximation Algorithm

The following algorithm demonstrates how an approximation factor (in this case a factor of 3 for metric instances) can be obtained based on LP-duality. We include its analysis as it is rather simple, compared to the one stated for instance for Algorithm 2.

Consider the following algorithm that consists of two phases. In the following, we give the algorithm as described by Vazirani [27, chap. 24]. The original version is found in [14].

Phase 1. Phase 1 uses a notion of time that allows associating events with the time at which these happened. Time starts at 0. Motivated by the desire to find a dual solution that is as large as possible, each client uniformly raises its dual variable α_j by 1 in unit time. Initially, all clients start at $\alpha_j = 0$ and all facilities are closed. There are two events that may happen:

Tight edge: When $\alpha_j = c_{ij}$ for some edge (i, j), the algorithm will declare this edge to be *tight*. After this event, if the facility i is not yet *temporarily open* (the event in which facilities are opened is described below), the dual variable β_{ij} corresponding to the tight edge will be raised at the same rate as α_j ensuring that the first dual constraint ($\alpha_j - \beta_{ij} \leq c_{ij}$) is not violated. For $\beta_{ij} > 0$ the edge (i, j) is called *special*. If facility i is already temporarily open, the client j is simply connected to i and stops raising its dual variables.

Facility paid for: Facility i is said to be *paid for* if $\sum_{j \in C} \beta_{ij} = f_i$. When this happens, the algorithm declares this facility *temporarily open*. Furthermore, all unconnected clients having tight edges to i are declared *connected*. After this, facility i is called the *connecting witness* of the just connected clients $\{j\}$ and the respective dual variables $\{\alpha_j\}$ and $\{\beta_{ij}\}$ are not raised any further.

Phase 1 completes after all clients are connected to a temporarily open facility. If several events happen simultaneously, they are processed in arbitrary order.

Notice how the events in Phase 1 are designed to satisfy complementary slackness conditions (2.5) and (2.6), respectively. The problem at the end of Phase 1, however, is that the client may have contributed to temporarily opening more than one facility (and thus too many facilities are temporarily open)[1].

Phase 2. Let F_t denote the set of temporarily open facilities and T denote the subgraph of G consisting of edges that were special at the end of Phase 1. Let T^2 denote the graph that has edge (u, v) iff there is a path of length at most 2 between u and v in T. Further let H be the subgraph of T^2 induced on F_t. Compute a maximal independent subset, I, of these vertices w.r.t. to the graph H. All facilities in set I are declared *open*.

Finally, each client j is connected to a facility in I as follows: If there is a tight edge (i, j) and facility i is open (i.e., $i \in I$), j is declared *directly connected* to i and the algorithm sets $\sigma(j) := i$ and $\beta_{i'j} := 0$ for all $i' \neq i$. Otherwise, consider the tight edge (i, j) such that i was the connecting witness for j. Since $i \notin I$, its closing witness, say i' must be in I. Thus, j is declared *indirectly connected* to i' and $\sigma(j)$ is set to i'.

The resulting *open* facilities and *connected* clients define a primal integral solution: That is, $y_i = 1$ iff $i \in I$ and $x_{ij} = 1$ iff client j is connected to i. Further, α_j and β_{ij} constitute a dual solution.

Please note how the algorithm is formulated in terms of the dual variables which then drive the execution and determine which facilities are opened and which clients are connected. Similar re-formulations in terms of the duals exist for example for Algorithm 2, which then opens up the way for its analysis [13]. Let us proceed to analyze the approximation ratio of this algorithm.

Analysis. The analysis relies on the following three points, of which the first two can be derived from the respective updates of the algorithm:

[1] Consider the complementary slackness condition (2.6) that each facility, which is open, should be fully paid for $\sum_{j \in C} \beta_{ij} = f_i$. Consider a facility i that has become temporarily open (because client j contributed β_{ij} to the above sum) *after* client j has been connected to a different facility (say i'). Thus, $y_i \neq x_{ij}$ but $\beta_{ij} > 0$. This violates condition (2.8), namely that $\beta_{ij} > 0 \Rightarrow y_i = x_{ij}$, and implies that in an optimal solution only one β_{ij} can be non-zero for a client j. These violations are repaired by closing some temporarily open facilities in Phase 2.

1. As a result of Phase 2, every client is connected to a facility and is only connected if this facility is also open. Therefore, y_i and x_{ij} are a feasible (and integral) solution of the primal ILP of eq.(2.2).
2. Similarly, the updates of the dual variables in the algorithm, never violate the dual constraints of eq. (2.3), and the resulting α_j and β_{ij} are a feasible solution for the dual LP.
3. The dual objective function bounds the primal objective function from above by a factor of 3. Note that the authors even show a stronger inequality (Theorem 2.3.5) which allows using this algorithm as a subroutine in an approximation of the k-Median problem.

Note that the algorithm satisfies all complementary slackness conditions for directly connected clients. For a client j *indirectly connected* to facility i (for which $\beta_{ij} = 0$), the respective complementary slackness condition $\alpha_j = c_{ij} + \beta_{ij}$ is relaxed to $\frac{1}{3}c_{ij} \leq \alpha_j \leq c_{ij}$.

For easier analysis, the authors split up the dual variables α_j into two sub-terms $\alpha_j = \alpha_j^e + \alpha_j^f$. By α_j^e they refer to the amount in α_j that is used towards paying for the connection costs to the facility $\sigma(j)$. By α_j^f they denote the amount in α_j that is used by client j to pay for opening a facility in I (j's part in the sum $\sum_{j \in C} \beta_{ij} = f_i$).

If i is indirectly connected, then $\alpha_j^f = 0$ and $\alpha_j^e = \alpha_j$. If j is directly connected to i, by definition of the algorithm, $\alpha_j = c_{ij} + \beta_{ij}$, that is, $\alpha_j^f = \beta_{ij}$ and $\alpha_j^e = c_{ij}$.

In the following, we give the proof as stated in [27, chap. 24]:

Lemma 2.3.2. *Let $i \in I$. Then,*

$$\sum_{j:\sigma(j)=i} \alpha_j^f = f_i.$$

Proof. The condition for declaring facility i temporarily open at the end of Phase 1 was that it is completely paid for, i.e,

$$\sum_{j:(i,j) \text{ is special}} \beta_{ij} = f_i.$$

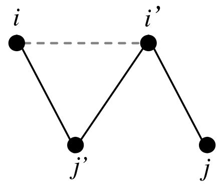
In Phase 2, as $i \in I$, all clients that have contributed to i in Phase 1 will be directly connected to i. For each such client, $\alpha_j^f = \beta_{ij}$. Any other client j' that is connected to facility i must satisfy $\alpha_{j'}^f = 0$. The lemma follows.

Corollary 2.3.3. $\displaystyle\sum_{i \in I} f_i = \sum_{j \in C} \alpha_j^f$.

Because $\alpha_j^f = 0$ for indirectly connected clients, only the directly connected clients pay for the cost of opening facilities. The sum of all these clients pays to open all facilities.

Lemma 2.3.4. *For a client j indirectly connected to a facility i, $c_{ij} \leq 3\alpha_j^e$.*

Proof. Let i' be the connecting witness for a client j. Since j is indirectly connected to i, (i, i') must be an edge in H. In turn, there must be a client, say j' such that (i, j') and (i', j') were both special edges at the end of phase 1. Let t_1 and t_2 be the times at which i and i' were declared temporarily open during Phase 1.



Since edge (i', j) is tight, $\alpha_j \geq c_{i'j}$. Below we will show that $\alpha_j \geq c_{ij'}$ and $\alpha_j \geq c_{i'j'}$. Then, the lemma will follow by using the triangle inequality[2]

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

Since edges (i', j') and (i, j') are tight, $\alpha_{j'} \geq c_{ij'}$ and $\alpha_{j'} \geq c_{i'j'}$. Because both these edges are special, they must both have gone tight before either i or i' is declared temporarily open. Consider the time $\min(t_1, t_2)$. Clearly, $\alpha_{j'}$ cannot be growing after this time, that is, $\alpha_{j'} \leq \min(t_1, t_2)$. Finally, since i' is the connecting witness for j, $\alpha_j \geq t_2$. Therefore, $\alpha_j \geq \alpha_{j'}$, and the required inequalities follow.

Theorem 2.3.5. *The primal and dual solutions constructed by the algorithm satisfy*

$$\sum_{i \in F,\ j \in C} c_{ij} x_{ij} + 3 \sum_{i \in F} f_i y_i \leq 3 \sum_{j \in C} \alpha_j \ .$$

Proof. For a client j directly connected to i, $c_{ij} = \alpha_j^e \leq 3\alpha_j^e$. Combining this with Lemma 2.3.4 one obtains

$$\sum_{i \in F,\ j \in C} c_{ij} x_{ij} \leq 3 \sum_{j \in C} \alpha_j^e \ . \tag{2.9}$$

The theorem results from adding the equation of Corollary 2.3.3 multiplied by 3 to the above (2.9).

Variant	**Lower bound**	**Approx. factor**	**Time compl.**	**Reference (first author)**
Set cover	$\Omega(\ln(\lvert U\rvert))$	$1+\ln(\lvert U\rvert)$	$\lvert U\rvert^3$	Chvátal [5]
General UFL	$\Omega(\ln(n))$	$1+\ln(n)$	$(n+m)^3$	Hochbaum [12]
Metric UFL	1.463 in [10]	3.16	LP	Shmoys [25]
		3	$nm\log(nm)$	Jain [14]
		2.41	LP	Guha [10]
		1.861	$nm\log(nm)$	Jain [13]
		1.61	$(n+m)^3$	Jain [13]
		1.52	$(n+m)^3$	Mahdian [20]

Fig. 2.4. Approximation algorithms for UFL. In the above notation, n denotes the number of clients and m denotes the number of facilities.

2.3.6 Summary

We conclude this section with a summary of known lower bounds and centralized approximation algorithms in Figure 2.4.

In the table, we noted Algorithm 1 by Hochbaum [12], the respective equivalent set-cover algorithm by Chvátal [5], the factor 3-approximation algorithm [14, 27] presented in Section 2.3.5, and Algorithm 2 by Jain et al. [13]. We also want to mention a second improved algorithm in the same paper [13] for which the authors prove an approximation ratio of 1.61.

The first known constant-factor approximation by Shmoys et al. [25] is also included in Figure 2.4 together with an LP-based algorithm by Guha and Khuller [10], who proved a lower bound of 1.463 for *metric* UFL. Moreover [10] and [13] were combined by Mahdian et al. [20] to obtain the currently best known approximation factor of 1.52 for this problem (also based on dual fitting). Note that the above list of centralized approximations and variants is not exhaustive, for further references refer to [20].

In the remainder of this chapter, we present distributed facility location algorithms which are built around the elements introduced in this section, most prominently the concept of opening facilities which have a good *cost-efficiency* – and the parallelization of this step.

2.4 Simple Distributed Approximation

In this section we discuss how Algorithm 2 can be executed in a distributed manner. Consider the distributed re-formulation we give in Algorithm 3 (facilities) and 4 (clients).

[2] This is where the analysis assumes a *metric* instance of UFL.

Please be aware that the distributed re-formulation still requires a *metric* instance which demands that the underlying bipartite graph is complete (or at least consists of cliques). In a graph derived from a multi-hop network, this implies global communication among nodes of a connected component. These algorithms are therefore not efficient for large multi-hop networks. Note that this restriction does not apply to the faster algorithm presented in Section 2.5 as it does not require a metric instance.

Assume that after a neighbor discovery phase, each client j has learned the set of neighboring facilities, which it stores in the local variable F, and vice versa each facility i has learned the set of neighboring clients, which it stores in the local variable C. Moreover, after initialization, each client j knows its respective c_{ij} for all $i \in F$ and equally each facility i knows its respective c_{ij} for all $j \in C$.

Algorithm 3: Distributed Formulation of Algorithm 2 for Facility i

```
1  set $U = C$
2  **while** $U \neq \emptyset$ **do**
3      find most cost-efficient star $(i, B)$ with $B \subseteq U$
4      **send** $c(i, B)$ to all $j \in U$
5      **receive** "connect-requests" from set $B^* \subseteq U$.
6      **if** $B^* = B$ **then**
7          open facility $i$ (if not already open)
8          **send** "open" to all $j \in B$
9          set $\sigma(j) = i$ for all $j \in B$
10         set $U = U \setminus B$
11         set $f_i = 0$
12     **else**
13         **send** "not-open" to all $j \in B^*$
14     **receive** "connected" from set $B'$
15     set $U = U \setminus B'$
```

Algorithm 4: Distributed Formulation of Algorithm 2 for a Client j

```
1  **while** *not connected* **do**
2      **receive** $c(i, B)$ from all $i \in F$
3      $i^* = \text{argmin}_{i \in F}\, c(i, B)$ // *use node ids to break ties among equal* $c(i, B)$
4      **send** "connect-request" to $i^*$
5      **if** *received "open" from* $i^*$ **then**
6          set $\sigma(j) = i^*$
7          **send** "connected" to all $i \in F$
```

Note that for ease of exposition, we describe the two algorithms in terms of a synchronous message passing model [23] although the algorithms do not depend on it. In particular, in each round, facilities and clients perform one iteration of the while loop.

Each facility i maintains a set of uncovered clients U which is initially equal to C (line 1). Facilities start a round by finding the most cost-efficient star (i, B) with respect to U and sending the respective cost efficiency $c(i, B)$ to all clients in B (lines 3-4). In turn, the clients receive cost-efficiency numbers $c(i, B)$ from many facilities (line 2). In order to connect the most cost-efficient star among the many existing ones, clients reply to the facility i^* that has sent the lowest cost-efficiency value with a "connect request" (lines 3-4). Facilities in their turn collect a set of clients B^* which have sent these "connect requests" (line 5). Intuitively, a facility should only be opened if $B = B^*$, that is, if it has connect requests from all clients B in its most efficient star (line 6). This is necessary, as it could happen that some clients in B have decided to connect to a different facility than i as this facility spans a more cost-efficient star. So, if all clients in B are ready to connect, facility i opens, notifies all clients in B about this, removes the connected clients from B, and sets its connection costs to 0 (lines 7-11).

If a client receives such an "open" message from the same facility i^*, which it had previously selected as the most cost-efficient (line 5), it can connect to i^*. Further it notifies all other facilities $i \neq i^*$ that it is now connected, such that these facilities can update their sets of uncovered clients U (line 15) and start the next round.

Note that the distributed algorithms correctly implement Algorithm 2: Consider the globally most efficient star (i^*, B^*) selected by the centralized algorithm. In the distributed version, the clients in B^* will all receive the respective cost-efficiency from i^* and select i^* as the most cost-efficient facility – if more than one most cost-efficient facility exists, the clients agree on one of them, which has the smallest id. Therefore all clients in B^* will reply a "connect-request" to i^*, which will open facility i^*.

Thus, the distributed algorithms connect the most efficient star in each communication round. However, additionally, in the same round other facilities and associated stars may be connected. This is particularly the case if the instance graph is not complete, which often occurs in instances derived from ad-hoc and sensor networks, and results in these stars being connected *earlier* than in the centralized algorithm. Regarding such stars, we show below that, given any star (i, B) connected in round k and any client $j \in B$, no star (i^+, B^+) connected in a later round $k^+ > k$ exists such that $j \in B^+$ and $c(i^+, B^+) < c(i, B)$. This implies that for each (i, B), the centralized algorithm would either also connect (i, B) before it will connect any (i^+, B^+) or, if $c(i, B) > c(i^+, B^+)$, that (i, B) is completely disjoint from any (i^+, B^+) that has fallen behind compared to the centralized algorithm.

Assume the contrary, namely that such stars (i, B) and (i^+, B^+) exist with $j \in B$ and $j \in B^+$. Observe that the star (i^+, B^+) could have equally been connected in the current round k (by assumption, clients in B^+ are uncovered and facility i^+ is not open in round k^+; this also holds for round k). In round k, however, client j has sent its "connect-request" message to i as this is required for connecting (i, B) – which contradicts the assumption that $c(i^+, B^+) < c(i, B)$.

Note that similarly to the above re-formulation of Algorithm 20, also the improved version presented in the same paper giving a 1.61 approximation guarantee and the 3-approximation algorithm of Section 2.4 can be re-formulated for distributed execution.

However, analogously to the maximal independent set algorithm presented Section **??**, the worst-case number of rounds required by Algorithms 3 and 4 is $\Omega(m)$, because there can be a unique point of activity around the globally most cost-efficient facility i^* in each round: Consider for instance a chain of m facilities located on a line, where each pair of facilities is interconnected by at least one client, and assume that facilities in the chain have monotonously decreasing cost-efficiencies. Each client situated between two facilities will send a "connect-request" to only one of them (the more cost-efficient) and therefore the second cannot open. In this example, only the facility at the end of the chain can be opened in one round.

The linear number of rounds required in the worst-case is impracticable in many cases. Nevertheless, the constant approximation guarantees of 1.861 – or even 1.61 for a similar algorithm – inherited from the centralized versions are intruiguing and the *average* number of rounds required for typical instances in sensor networks remains to be explored. A much faster distributed algorithm, which runs in a constant number of rounds and can additionally handle arbitrary (non-metric) instances, is presented in the following Section 2.5.

2.5 Fast Distributed Approximation

Moscibroda and Wattenhofer [22] present the first distributed algorithm that provides a non-trivial approximation of UFL in a number of rounds that is *independent of the problem instance*. The approach follows the line of a number of recent approximation algorithms, e.g. [17], that employ an integer linear program (ILP) formulation of the addressed optimization problem, solve its linear relaxation in a distributed manner, and finally employ schemes to convert the fractional results to an integer solution with approximation guarantees.

The algorithm consists of two phases. In the first phase, given an arbitrary integer k, the algorithm computes an $O(\sqrt{k}(m\rho)^{1/\sqrt{k}})$ approximation of the LP relaxation in $O(k)$ communication rounds. In this bound, ρ is a factor depending on the coefficients c_{ij} and f_i and further it is required that $c_{ij} \geq 1$ and $f_i \geq 1$. To dispose of the dependency on ρ in the approximation ratio and to allow arbitrary coefficients, the authors give a transformation algorithm that runs in $O(k^2)$ rounds and yields an approximation ratio of $O(k(mn)^{1/k})$ [22]. Then in a second phase, the fractional result is rounded into a feasible integer solution of the ILP, worsening the approximation ratio only by a factor of $\log(m+n)$. The authors provide distributed algorithms that can be executed on facility and client nodes, respectively, for both phases of the algorithm. Because the algorithms – especially of the first phase – are quite intricate, in the following we will omit details on how the dual variables are updated. These updates, based on *dual fitting* (see Section 2.3.5), are designed to implement the required complementary slackness conditions, while allowing the dual to become infeasible – but only slightly, such that the dual variables divided by a certain factor yield a feasible dual solution.

Note that the presented approach is, in the context of distributed LP approximation [2, 19], the first to approximate a linear program (namely equation (2.2)) which is not a *covering or packing* LP, i.e., a linear program in which negative coefficients are allowed.

2.5.1 LP Approximation

In order to obtain an approximate solution of the relaxed linear program UFL_{frac}, facilities and clients execute Algorithms 5 and 6, respectively. The algorithms perform stepwise parallel increases of the variables x_{ij} and y_i, which start at zero. Each facility i will increase the variable y_i determining whether facility i should be open, while each client j will increase its connection variables x_{ij} determining to which facilities i it is connected. These increases are made in k synchronous parallel steps, determined by an outer loop (line 4) and a nested inner loop (line 5) which are each executed $\lceil\sqrt{k}\rceil$ times. Note that a synchronous message passing model as in [23] is assumed; the authors state that a synchronizer [1] can be used to make the algorithm work in asynchronous settings.

In each round, the *clients* compute whether they are *covered*, i.e., whether they are fractionally connected to a facility ($\sum_{i\in F} x_{ij} \geq 1$), and send this information to all facilities in line 6. Each *facility* receives this information and computes the set $U$of *uncovered* clients in lines 6 and 7.

The *facility* algorithm will then perform parallel increases of the y_i variables. The choice of which of the y_i to increase is based – as in the centralized Algorithms 1 and 2 presented above – on how beneficial a facility is to the set $U$of yet *uncovered* clients: A *facility* node i will increase the respective y_i if there exists a star (i, B) with $B \subseteq U$ that has a cost efficiency[3] smaller than a step-wise increasing threshold $\rho^{s/h}$ (line 8). Note that the threshold $\rho^{s/h}$ depends on the outer s-loop (an exemplary increase of $\rho^{s/h}$ with s is shown in Figure 2.6(a), we will discuss its function later). If such a star exists, a facility computes the *largest* such star (i, B_i') which still has a cost efficiency that is less than $\rho^{s/h}$ (line 9). Note that the set B_i' can simply be

[3] We refer to the cost efficiency as defined in eq. (2.1).

Algorithm 5: LP Approximation - Facility i

```
1  $h := \lceil\sqrt{k}\rceil$
2  $\rho := \max_{j\in C}\min_{i\in F}(c_{ij} + f_i)$
3  $y_i := 0;\ U := C$
4  for $s := 1$ to $h$ by $1$ do
5      for $t := h-1$ to $0$ by $-1$ do
6          receive "covered" from subset $B^- \subseteq C$
7          $U := U \setminus B^-$
8          if $\exists$ star $(i, B)$ with $B \subseteq U$ and $c(i,B) \leq \rho^{s/h}$ then
9              find largest star $(i, B_i')$ with $B_i' \subseteq U$ and $c(i, B_i') \leq \rho^{s/h}$
10             if $y_i < m^{-t/h}$ then
11                 raise $y_i$ to $m^{-t/h}$
12                 send $y_i$ to all $j \in B_i'$
13         update duals (omitted)
```

Algorithm 6: LP Approximation - Client j

```
1  $h := \lceil\sqrt{k}\rceil$
2  send $c_{ij}$ to all $i \in F$
3  $\forall i \in F : x_{ij} := 0$
4  for $s := 1$ to $h$ by $1$ do
5      for $t := h-1$ to $0$ by $-1$ do
6          if $\sum_{i\in F} x_{ij} \geq 1$ then send "covered" to all $i \in F$
7          receive $y_i$ from all $i \in F$ with $j \in B_i'$
8          forall $i \in F$ with $j \in B_i'$ do
9              if $\sum_{l\in F} x_{lj} < 1$ then
10                 $x_{ij} := y_i$
11         update duals (omitted)
```

constructed as $B_i' := \{j|c_{ij} \leq \rho^{s/h}\}$, see the original paper for details and proof [22]. Further, a facility i increases its y_i to a new level $m^{-t/h}$ in line 11. This *raise level* $m^{-t/h}$ depends on the inner t-loop and increases up to 1 in the last iteration of the loop (see Figure 2.6(a) for an example). Facility i then notifies all clients in its largest star B_i' of its increase of y_i in line 12.

The corresponding *client* algorithms then receive the respective y_i from such facilities i in line 7. Clients then increase their x_{ij} to y_i in line 10 unless they have become covered earlier. This means that a client j connects itself (fractionally) to all facilities i which – in this step – increased y_i and thus have become "more open" in the fractional sense.

2.5.2 Example

In the following, we sketch an exemplary execution of Algorithms 5 and 6 using an instance as shown in Figure 2.5. It shows a network of three facilities and four clients with respective opening and connection costs.

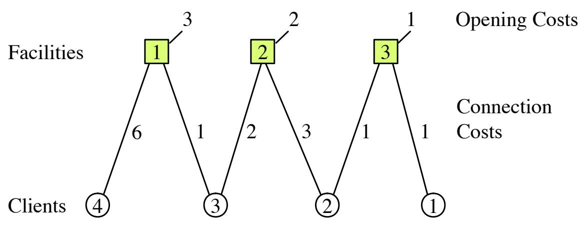


Fig. 2.5. An example instance with three facilities and four clients

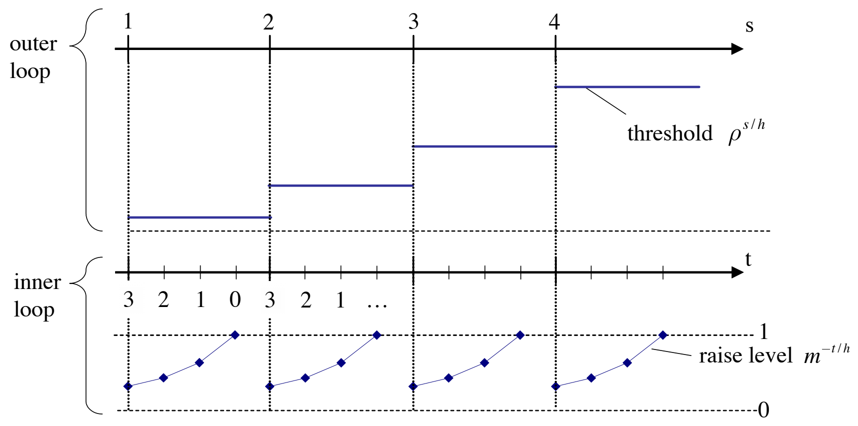
The actual execution of the algorithms is shown in Figure 2.6. In this example, we let the algorithms perform a total of $k = 16$ steps, resulting in $h = \sqrt{k} = 4$ steps in each loop. We show the two values that drive the algorithm's execution, the *efficiency threshold* $\rho^{s/h}$ and the *raise level* $m^{-t/h}$, in Figure 2.6(a).

The *efficiency threshold* is important for parallelization of the central greedy step of the centralized approximation algorithms in Section 2.3 above (opening the facility that constitutes the most cost-efficient star). Figure 2.6(a) shows an example of how the threshold is increased from $\rho^{1/h}$ up to ρ. The idea of the outer s-loop is to increase the y_i of facilities with comparably good cost efficiency and further connect the respective clients. In its last step, an efficiency threshold of ρ is high enough to include all not yet connected clients in the last step (see also Lemma 2.5.1). In the chosen instance of Figure 2.5, $\rho = 9$. That is, in the last step, client 4 can be connected by facility 1 forming a single-client star with a cost-efficiency of 9, while all other clients can be connected (possibly earlier) via stars that are more cost-efficient.

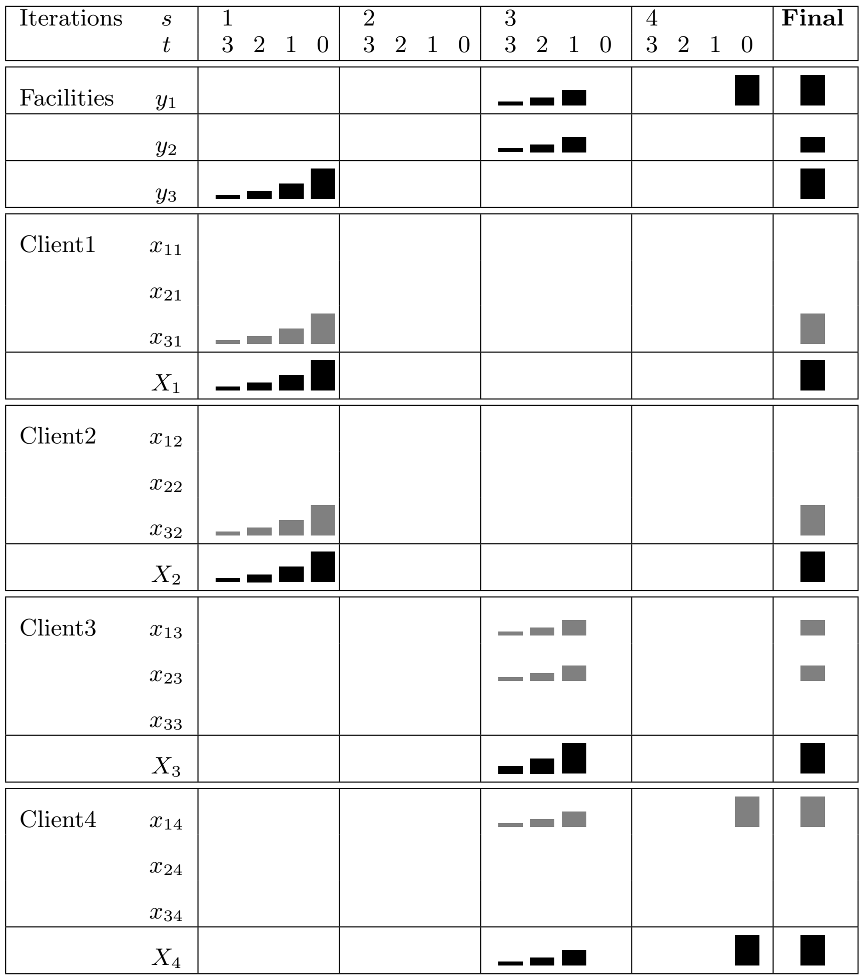
In Figure 2.6(b), we give the respective variables y_i and x_{ij}. By X_j we refer to the sum $\sum_i x_{ij}$ of all connections of client j. We use X_j to emphasize whether the client j is already *covered* or still active.

Consider the first step of the s-loop ($s = 1$). At this time, only facility 3 can form a star with cost efficiency smaller than $\rho^{1/4} \approx 1.73$. In this case, clients 1 and 2 are part of this star, which is also facility 3's largest star B'_3 with *cost efficiency* less than $\rho^{1/4}$ in line 9 of the *facility* algorithm. The inner t-loop then raises y_3 and sends this update to all $j \in B'$. Clients 1 and 2 thus set x_{31} and x_{32} to the same amount in line 10.

The *raise level* $m^{-t/h}$, also shown in Figure 2.6(a), denotes the value to which the variables y_i are raised in each step of the inner t-loop (the respective



(a) Efficiency threshold $\rho^{s/h}$ and raise level $m^{-t/h}$. The outer s-loop and the nested t-loop each perform $h = \sqrt{k} = 4$ steps.



(b) Facility and client variables during execution. Full/missing bars indicate 1 or 0, respectively. Bars are omitted if no raise occurs.

Fig. 2.6. Exemplary execution of Algorithms 5 and 6. We show an instance of three facilities and four clients. The number of rounds k is set to 16.

x_{ij} are raised to the same level unless already covered). In this example the inner loop is performing four steps from $m^{-3/4}$ to $m^{0/4} = 1$. Thus, in the final step of the t-loop, the clients 1 and 2 become *covered* as they are connected to facility 3. Thus, clients 1 and 2 are removed from the set of uncovered clients U and are not considered further in the computation of stars at later steps. Because of this, the cost efficiency of the most efficient star is always increasing at all facilities (proof omitted).

In the next step of the outer loop ($s = 2$), no facility is efficient enough in connecting the clients remaining in U(i.e., can form a star with efficiency $\leq \rho^{2/4} = 3$ with clients 3 and/or 4), therefore nothing happens.

In the next iteration of the outer loop ($s = 3$), the *efficiency threshold* is increased again to $\rho^{3/4} \approx 5.2$. As facilities execute their algorithms in parallel, at this time both facilities 1 and 2 have found stars with efficiency levels $c(i, B)$ that are below the threshold. In this example, facility 1 can connect the client set $\{3, 4\}$ with cost efficiency $c(1, \{3, 4\}) = 5$ and facility 2 can connect the client 3 with efficiency $c(2, \{3\}) = 4$. Note that both facilities have the client 3 in their respective *largest stars*, B'_1 and B'_2. In this setting, the role of the inner t-loop becomes evident, namely to ensure that the sum $\sum_{i \in F} x_{ij}$ is not increased too much if there are many facilities that simultaneously increase their y_i and thus cause an increase of many x_{ij} of a given client j.

In the given example, in each step of the inner t-loop, both facilities increase their respective y_i, and each facility then causes an update of the respective x_{ij}, that is, facilities 1 and 2 contribute to increases of x_{13} and x_{23}, respectively, and facility 1 additionally contributes to increases of x_{14}. Therefore, as X_3 is increased twice in each step, the client already becomes (fractionally) covered by facilities 1 and 2 in the second-to-last step of the inner loop. In the last t-step, client 3 is already covered, and client 4 is the only client remaining in U, with which neither facility can form an efficient-enough star, therefore client 4 is not covered any further.

In the final step of the outer loop ($s = 4$), all yet (fractionally) uncovered clients (here client 4) will be connected, as the *efficiency threshold* is set to ρ. In this example, the increases of the inner loop only have an effect after the *raise level* is above the actual y_1, which is in the very last step. In this last step of the inner t-loop, y_1 and x_{41} are both raised to 1.

2.5.3 Analysis

In the following, we make a few observations about Algorithms 5 and 6. For brevity, we omit large parts of the proof. These can be readily found in [22].

Lemma 2.5.1. *Algorithms 5 and 6 produce a feasible solution for UFL*$_{\text{frac}}$.

Proof. The feasibility of the second LP constraint, $x_{ij} \leq y_i$ for all $i \in F$ and $j \in C$, directly follows from the algorithm definition in lines 11, 12 and 7, 10 of the facility and client algorithms, respectively. The increase of a connection variable Δx_{ij} never exceeds the previous increase of the corresponding Δy_i.

As for the first LP constraint, $\sum_i x_{ij} \geq 1$ for all $j \in C$, assume for contradiction that j is a client which is still *uncovered* at the end of the algorithm, i.e., $\sum_i x_{ij} < 1$ and $j \in U$. 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 forming a star $(i, \{j\})$ with cost efficiency $c(i, \{j\}) \leq \rho$[4]. Because $s = h$, facility i will increase its variable to $m^{-t/h} = 1$ in line 11. Subsequently, j will set $x_{ij} = 1$ in line 10, which contradicts the assumption that $\sum_i x_{ij} < 1$ at the end of the algorithm.

Further, the algorithm achieves approximation guarantees through a careful handling of the dual variables α_j and β_{ij} of the corresponding dual LP (eq. (2.3)). Essentially, the method of *dual fitting* is applied in a distributed setting. While we omit the details of the dual updates in line 11 of Algorithm 5 and line 13 of Algorithm 6, we highlight the following property of these algorithms:

Lemma 2.5.2. *At the end of each t-loop, the primal objective function $\sum_{j \in C, i \in F} c_{ij} x_{ij} + \sum_{i \in F} f_i y_i$ is equal to the dual objective function $\sum_{j \in C} \alpha_j$.*

Furthermore, the algorithm ensures that dual infeasibility is bounded by a factor γ, and thus in a final step, the dual variables can be divided by γ to yield a feasible dual solution. For details we refer to the original paper [22]. Because of basic laws of LP-duality, the primal solution is then a γ-approximation of UFL_{frac}.

This leads to the approximation guarantee of Algorithms 5 and 6.

Theorem 2.5.3. *For an arbitrary integer $k > 0$ algorithms 5 and 6 compute a $O(\sqrt{k}(m\rho)^{(1/\sqrt{k})})$ approximation of UFL_{frac}.*

For this bound it is required that $c_{ij} \geq 1$ and $f_i \geq 1$. In a re-formulation that is independent of ρ and allows arbitrary coefficients, the algorithm uses a scaling technique that requires $O(k^2)$ steps. In this case the approximation factor becomes $O(k(mn)^{1/k})$. We will discuss implications of the obtained approximation in Section 2.5.5.

2.5.4 Rounding

So far, the solution computed in the first phase of the algorithm is fractional, i.e., it solves UFL_{frac}without requiring the variables y_i and x_{ij} to be integers. A second rounding phase, shown in Algorithm 7 and Algorithm 8, is therefore used to construct a feasible integer solution for a UFL instance. For this, each *facility* i sets its y_i to 1 with probability p_i and to 0 otherwise, and sends y_i

[4] Consider the definition of $\rho = \max_{j \in C} \min_{i \in F}(c_{ij} + f_i)$ in line 2 and let i be the facility for which the term $\min_{i \in F}(c_{ij} + f_i)$ is minimal for the uncovered client j. The cost efficiency of the star $(i, \{j\})$ is at most $c_{ij} + f_i$. By the definition of $\rho = \max_{j \in C}(\dots)$, the cost efficiency of this star is less or equal to ρ.

to all clients. Later, if they receive a JOIN message from any client, they set their y_i to 1.

Each *client* computes its current contribution to the fractional service costs C_j^* (line 1) and the set V_j of facilities within a $\ln(n+m)$ factor of C_j^* (line 2). Then each j receives y_i from all facilities i and defines its neighborhood N_j as the open facilities in V_j. If at least one such neighboring facility is available, the open facility with least service cost is chosen (line 7) or, otherwise, the closest facility is sent a JOIN message in line 11.

Algorithm 7: Rounding - Facility i

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

Algorithm 8: Rounding - Client j

```
1  $C_j^* := \sum_{i \in F} c_{ij} \hat{x}_{ij}$
2  $V_j := \{i \in F | c_{ij} \leq \ln(n+m) \cdot C_j^*\}$
3  **receive** $y_i$ from all $i \in F$
4  $N_j := V_j \cap \{i \in F | y_i = 1\}$
5  **if** $N_j \neq \emptyset$ **then**
6      $i' := \text{argmin}_{i \in F}\, c_{ij}$
7      $x_{i'j} := 1$
8  **else**
9      $i' := \text{argmin}_{i \in F}(c_{ij} + f_i)$
10     $x_{i'j} := 1$
11     **send** JOIN message to facility $i'$
```

Theorem 2.5.4. *Let $\hat{x}_{ij}, \hat{y}_i$ for each $j \in C$ and $i \in F$ denote the fractional solution with cost at most $\gamma \times OPT$ obtained by Algorithms 5 and 6. In two rounds of communication, Algorithms 7 and 8 produce an integer solution x_{ij}, y_i with cost at most $O(\log(m+n))\gamma OPT$ in expectation.*

Proof. It follows from the definition of C_j^* and V_j that $\sum_{j \in F \setminus V_j} \hat{x}_{ij} \leq 1/\log(n+m)$, because if not, C_j^* would be larger. The design of Algorithms 5 and 6 guarantees the invariant $\sum_{i \in V_j} \hat{x}_{ij} = \sum_{i \in V_j} \hat{y}_i$ and $\sum_{i \in F} \hat{x}_{ij} \geq 1$, thus it holds that

$$\sum_{i \in V_j} \hat{y}_i = \sum_{i \in V_j} \hat{x}_{ij} \geq 1 - \frac{1}{\log(n+m)}. \tag{2.10}$$

For each client j having $N_j \neq \emptyset$, the connection costs c_{ij} are at most $\log(n+m) \cdot C_j^*$ by the definition of the neighborhood V_j. It follows that these clients account for total connection costs of at most $\log(n+m) \sum_{j \in C, i \in F} c_{ij}\hat{x}_{ij}$. A facility declares itself *open* in line 2 of Algorithm 7 with probability $\min\{1, \hat{y}_i \cdot \log(n+m)\}$. The expected opening costs of facilities opened in line 2 are thus bounded by the value $\log(n+m) \sum_{i \in F} \hat{y}_i f_i$.

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

$$\begin{aligned}
q_j &= \prod_{i \in V_j} (1 - p_i) = \left(\sqrt[n+m]{\prod_{i \in V_j} (1 - p_i)} \right)^{n+m} \\
&\leq \left(\frac{\sum_{i \in V_j} (1 - p_i)}{n+m} \right)^{n+m} \\
&\overset{|V_j| \leq m}{\leq} \left(1 - \frac{\ln(n+m) \sum_{i \in V_j} \hat{y}_i}{n+m} \right)^{n+m} \\
&\overset{\text{Eq.2.10}}{\leq} \left(1 - \frac{\ln(n+m)}{n+m} \left(1 - \frac{1}{\ln(n+m)} \right) \right)^{n+m} \\
&= \left(1 - \frac{\ln(n+m) - 1}{n+m} \right)^{n+m} \\
&\leq e^{-\ln(n+m)-1} \leq \frac{1}{e(n+m)}.
\end{aligned}$$

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 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 represent a feasible solution otherwise. Combining this with the above results, the total expected cost $\mu = E[ALG]$ is

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

This concludes the proof of Theorem 2.5.4.

The above rounding algorithm obtains a solution that holds in expectation. If one is interested in results that hold with high probability, it is possible to change Algorithm 7 to obtain the above approximation with probability $1 - n^{-1}$. However, this requires additional coordination of the opening of facilities by a single leader node.

2.5.5 Obtainable Approximation

Note that the resulting approximation ratio of $O(\log(m+n) \times \sqrt{k}(m\rho)^{1/\sqrt{k}})$ deserves further discussion. Generally, the algorithm permits an arbitrary number of steps k. Against intuition however, the approximation factor cannot be improved arbitrarily with increasing k. In contrast, for $k \to \infty$ the approximation ratio even tends towards ∞.

That is, for any given instance size, there exists an optimal number of rounds k_{opt} for which the approximation factor is minimal. After more than k_{opt} steps, the approximation ratio degrades again, consequently it only makes sense to chose $k \in [1, k_{\text{opt}}]$. The best approximation guarantee is obtained for $k_{\text{opt}} = \log^2(m\rho)$ or, respectively in the formulation independent of ρ, $k_{\text{opt}} = \log^2(mn)$. This "point of operation" is interesting when comparing the algorithm to previous centralized algorithms and lower bounds shown in Figure 2.4.

Note that at k_{opt} rounds of execution, the last term of the approximation ratio, $(m\rho)^{1/\sqrt{k}}$, becomes a constant[5] which may be neglected in O-notation. The algorithm therefore obtains a $O(\log(m+n)\log(m\rho))$ approximation in $O(k_{\text{opt}} = \log^2(m\rho))$ steps.

Similarly, the best approximation ratio that is independent of ρ can be computed. In Figure 2.7, we give an overview of the best achievable approximation factors using an optimal number of rounds k_{opt}, together with the slow but simple distributed algorithm for metric instances given in Section 2.4.

Variant	Approximation Factor	Comm. Rounds
Distributed metric UFL	1.861 (or 1.61)	m
Distributed general UFL		
– dependent on ρ	$\log(m+n)\sqrt{k}(m\rho)^{1/\sqrt{k}}$	k
	$\log(m+n)\log(m\rho)$	$k_{\text{opt}} = \log^2(m\rho)$
– independent of ρ	$\log(m+n)\sqrt{k}(mn)^{1/\sqrt{k}}$	k
	$\log(m+n)\log(mn)$	$k_{\text{opt}} = \log^2(mn)$

Fig. 2.7. Trade-off between approximation guarantee, number of communication rounds, and dependency on ρ

[5] The equation $c = (m\rho)^{1/\log_a(m\rho)}$ can be solved by applying $\ln(\dots)$ to $c = (m\rho)^{\ln a/\ln(m\rho)}$ which yields $\ln c = \ln a$, that is, c is the basis of the logarithm.

Summing up, the distributed algorithm for *general* UFL instances described in this section provides the first distributed approximation whose running time is independent of the instance size. While the algorithm is not as flexible as to allow to come arbitrarily close to the centralized lower bound, when it is executed for a logarithmic number of rounds, its approximation ratio is within a logarithmic factor of the centralized lower bound.

In contrast, for the *metric* case, we have presented a (some will argue prohibitively slow) distributed version of the centralized greedy Algorithm 2, which in exchange provides a much better approximation ratio (Section 2.4). Note that such a distributed algorithm can be parallelized further, for instance by employing Luby's classic algorithm for computing a maximal independent set [18] (also described in Section **??**). This technique would be particularly suitable to implement a fast distributed version of the centralized 3-approximation algorithm described in Section 2.3.5.

2.6 Discussion and Outlook

We presented a simple distributed algorithm for metric instances of the facility location problem in Section 2.4. Further, we described a fast distributed algorithm [22] for general instances in the previous Section 2.5. Both were not particularly designed for the models applicable to multi-hop sensor networks. The algorithms operate on bipartite graphs, in which clients communicate with all eligible facilities in each round. The first algorithm, although distributed, preserves its approximation factor only with global communication among all nodes – as it requires a metric instance which implies that the underlying bipartite graph is complete. The second algorithm can be used for settings in which facilities and clients are direct neighbors. For example, it could be used to compute an energy-efficient clustering (Figures 2.1(b) and 2.3(a)) as such scenarios can be unfolded into bipartite graphs in which some edges are missing.

Notice that each of the algorithms has additional disadvantages even in this one-hop setting. The first algorithm for metric instances can be prohibitively slow, although we have noted that there is potential for accelerating a similar algorithm. The second algorithm, while in turn very fast and even suitable for general instances, requires that the coefficient ρ – computed from the costs of the respective instance – is known at all nodes. Note that computing and distributing ρ is inherently non-local. Nevertheless, in some settings, ρ can be determined in advance, e.g., prior to deployment. Also, adaptations could be provided to make the presented algorithm independent of ρ.

Further, for applications in which clients may connect to facilities that lie more than one hop away, both algorithms are not efficient, as each round requires "global" communication between eligible clients and facilities.

However, the presented approaches can well be adapted to these multi-hop settings: Notice that the main idea of all presented algorithms is to open – either sequentially or in parallel – the facilities $\{i\}$ that span the most *cost-efficient* stars $\{(i, B)\}$. We noted above that in spite of an exponential number of sets of uncovered clients $B \subseteq U$, a facility can find such a star by considering the clients ordered by the respective connection costs c_{ij} to facility i. In most multi-hop models, the clients are already ordered by c_{ij}, as c_{ij} will reflect a measure of network proximity between i and j. That is, a facility can consider clients in the order of their network proximity to itself. As a consequence, the clients in a least-cost star of a given size, will always be in a local network neighborhood around a potential facility i, as sketched in Figure 2.8.

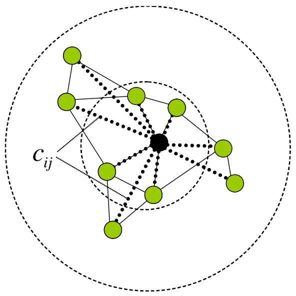


Fig. 2.8. A star in the multi-hop model

Similarly to the *efficiency threshold* – which determines which stars are connected in a given step of the fast distributed algorithm – the *scope* around each potential facility i could be increased, thus raising the connection variables x_{ij} at close clients at first and at more remote clients later.

This should enable approximating different variants of the facility location problem even in the multi-hop model. For example, one could stop the algorithm after a given step k and solve facility location for the case that clients and facilities can be at most *scope*(k) apart. Moreover, if the scope is doubled at each step, one may obtain an approximation for arbitrary distances between facilities and clients.

Furthermore, some wireless sensor network applications may only require a special case of UFL, for which better guarantees can be given. We have mentioned that most instances in multi-hop networks will be metric. Further, the costs of executing a service at a given node – constituting the opening costs of a facility – could be assumed to be known and equal at all nodes. Similarly, a fault tolerant version of UFL could be applied, in which each client j must be connected to at least r_j facilities. The fault-tolerant case is also made easier, if the redundancy requirement r_j is the same at all nodes.

2.7 Chapter Notes

We close with a few notes on further reading. Thorup [26] describes a method for improving the centralized runtime both of the presented Algorithm 2 and of an advanced version [13] that provides a 1.61 approximation factor. Chudak et al. [4] have discussed how the centralized algorithm of Section 2.3.5 [14] can be adapted to obtain a fast distributed algorithm which solves a constrained version of the problem requiring facilities and associated clients to be at most three network hops apart. Recent work [9] discusses a constant-time distributed algorithm for a constrained version of the problem in which opening costs at all nodes are equal. A distributed algorithm based on hill-climbing [16] addresses a version of the problem in which exactly k facilities are opened. Although in [16] the worst-case time complexity and the obtained approximation factor are not discussed explicitly, the approach is expected to adapt well to instances which change over time. Finally, in current work, we give more involved versions of Algorithms 3 and 4, which can be implemented in multi-hop networks and inherit the approximation factor of 1.61 from [13]. Although in theory requiring a high worst-case runtime, in practice these reformulations terminate in few communication rounds and require only local communication confined to small network neighborhoods.

1. Baruch Awerbuch. Complexity of network synchronization. *Journal of the ACM*, 32(4):804–823, 1985. 2.5.1
2. Yair Bartal, John Byers, and Danny Raz. Global optimization using local information with applications to flow control. In *Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science (FOCS'97)*, page 303, October 1997. 2.5
3. Alberto Cerpa and Deborah Estrin. ASCENT: Adaptive Self-Configuring Sensor Networks Topologies. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, 2002. 2.1
4. Fabian Chudak, Thomas Erlebach, Alessandro Panconesi, and Mauro Sozio. Primal-dual distributed algorithms for covering and facility location problems. Unpublished Manuscript, 2005. 2.7
5. Vašek Chvátal. A greedy heuristic for the set-covering problems. *Operations Research*, 4(3):233–235, 1979. 2.3.2, 2.3.2, 2.3.2, 2.3.6, 2.3.6
6. Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*, pages 455–462, Washington, DC, USA, 2004. IEEE Computer Society. 2.1
7. Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998. 2.3.1
8. Christian Frank and Kay Römer. Algorithms for generic role assignment in wireless sensor networks. In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SENSYS'05)*, San Diego, CA, USA, November 2005. 2.2.2
9. Joachim Gehweiler, Christiane Lammersen, and Christian Sohler. A distributed $O(1)$-approximation algorithm for the uniform facility location problem. In *Proceedings of the 18th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'06)*, pages 237–243, Cambridge, Massachusetts, USA, 2006. 2.7
10. Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31:228–248, 1999. 2.3.6, 2.3.6
11. Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, and Mani B. Srivastava. A dynamic operating system for sensor nodes. In *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services (MOBISYS'05)*, pages 163–176, New York, NY, USA, 2005. ACM Press. 2.1
12. Dorit S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22(1):148–162, 1982. 2.3.2, 2.3.6, 2.3.6
13. Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijai V. Vazirani. Greedy facility location algorithms analyzed using dual fitting

with factor-revealing LP. *Journal of the ACM*, 50:795–824, November 2003. 2.3.3, 2.3.4, 2.3.5, 2.3.6, 2.3.6, 2.7

14. Kamal Jain and Vijai V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS'99)*, pages 2–13, October 1999. 2.3.4, 2.3.5, 2.3.6, 2.3.6, 2.7
15. Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer-Verlag, 2 edition, 2002. 2.3.2
16. Denis Krivitski, Assaf Schuster, and Ran Wolff. A local facility location algorithm for sensor networks. In *Proceedings of the 1st International Conference on Distributed Computing in Sensor Systems*, Marina del Rey, CA, USA, June 2005. 2.7
17. Fabian Kuhn and Roger Wattenhofer. Constant-time distributed dominating set approximation. In *Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC'03)*, pages 25–32, Boston, MA, USA, 2003. ACM Press. 2.5
18. Michael Luby. A simple parallel algorithm for the maximal independent set problem. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing (STOC'85)*, pages 1–10, New York, NY, USA, May 1985. ACM Press. 2.5.5
19. Michael Luby and Noam Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC'93)*, pages 448–457, New York, NY, USA, May 1993. ACM Press. 2.5
20. Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Improved approximation algorithms for metric facility location problems. In *Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX'02)*, volume 2462 of *Lecture Notes in Computer Science*, pages 229 – 242, Rome, Italy, September 2002. 2.3.6, 2.3.6
21. Pitu B. Mirchandani and Richard L. Francis, editors. *Discrete Location Theory*. Discrete Mathematics and Optimization. Wiley, 1990. 2.1
22. Thomas Moscibroda and Roger Wattenhofer. Facility location: Distributed approximation. In *Proceedings of the 24th ACM Symposium on Principles of Distributed Computing (PODC'05)*, pages 108–117, New York, NY, USA, July 2005. ACM Press. 2.5, 2.5.1, 2.5.3, 2.5.3, 2.6
23. David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications, 2000. 2.4, 2.5.1
24. R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC'95)*, pages 475–484, 1997. 2.3.1
25. David B. Shmoys, Éva Tardos, and Karen Aardal. Approximation algorithms for facility location problems (extended abstract). In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC'95)*, pages 265–274, New York, NY, USA, 1997. ACM Press. 2.3.6, 2.3.6
26. Mikkel Thorup. Quick and good facility location. In *Proceedings of the 14th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'03)*, Baltimore, MD, USA, January 2003. 2.7
27. Vijai V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2 edition, 2003. 2.2.1, 2.3.4, 2.3.5, 2.3.5, 2.3.6
28. Jens Vygen. Approximation algorithms for facility location problems. Technical Report 05950-OR, Research Institute for Discrete Mathematics, University of Bonn, 2005. 2.1, 2.3.1, 2.3.4