Distributed Facility Location Algorithms for Flexible Configuration of Wireless Sensor Networks

Christian Frank and Kay Römer

Department of Computer Science
ETH Zurich, Switzerland
{chfrank,roemer}@inf.ethz.ch

Abstract. Many self-configuration problems that occur in sensor networks, such as clustering or operator placement for in-network data aggregation, can be modeled as facility location problems. Unfortunately, existing distributed facility location algorithms are hardly applicable to multi-hop sensor networks. Based on an existing centralized algorithm, we therefore devise equivalent distributed versions which, to our knowledge, represent the first distributed approximations of the facility location problem that can be practicably implemented in multi-hop sensor networks with local communication. Through simulation studies, we demonstrate that, for typical instances derived from sensor-network configuration problems, the algorithms terminate in only few communication rounds, the runtime does not increase with the network size, and, finally, that our implementation requires only local communication confined to small network neighborhoods. In addition, we propose simple extensions to our algorithms to support dynamic networks with varying link qualities and node additions and deletions. Using link quality traces collected from a real sensor network deployment, we demonstrate the effectiveness of our algorithms in realistic multi-hop sensor networks.

1 Introduction

An important problem in wireless sensor networks [1] is *self-configuration* [2], where network nodes take on different functions to achieve a given application goal. One example is *clustering* [3], where some nodes are elected as *cluster leaders*, serving as communication hubs for nearby nodes. A similar problem is aggregator placement [4], where some nodes are elected as *aggregators* that collect and aggregate sensor data from nearby sensor nodes. Recently, tiered sensor networks [5] have been proposed, consisting of resource-poor sensor nodes in the first tier and powerful hub nodes in the second tier. In these networks, every sensor node is assigned to and controlled by a hub node. Note that in all of the above examples, self-configuration consists in electing some nodes as *servers* while the remaining *client* nodes are assigned to a server.

While many proposals exist for finding such network configurations, they often do not pay attention to optimizing the *overall cost* of these configurations, which consists of two components: on the one hand, the costs of operating the servers (e.g., representing the servers' increased communication load as these forward traffic for many clients), and, on the other hand, the costs of communication between clients and their server. In wireless networks, the latter cost can be dependent on the physical distance between a client and its server (as a longer wireless link requires higher transmit power and thus increased energy consumption), on the number of hops in a multi-hop network graph, or on interference and network congestion. In all cases, lowering communication costs by means of additional hub nodes may prove beneficial.

Our goal is the provision of a *generic* and *practical* mechanism for finding cost-optimized solutions to the above self-configuration problems. Our approach is based on the observation that the above optimization problem can be modeled as an (uncapacitated) *facility location problem*. There, we are given a set F of *facilities*, a set C of *clients* (also known as cities or customers), a cost f_i for opening a facility $i \in F$ and connection costs c_{ij} for connecting client j to facility i. The objective is to open a subset of facilities in F and connect each client to an open facility such that the sum of connection and opening costs is minimized.

Although the facility location problem has been studied extensively in the past, no *practical* solutions exist that would be suitable for multi-hop sensor networks. While distributed algorithms for facility location exist, they are either not generally applicable [6], require a certain (albeit small) amount of global knowledge [7], require impractical communication models [7,8], or (based on the provided approximation factor [8]) might not improve over existing configuration heuristics for sensor networks.

We therefore contribute a local facility location algorithm that lends itself well for implementation in multi-hop sensor networks and provides an approximation factor of 1.61 for metric instances. By means of an experimental study, we show that the algorithm terminates after few communication rounds for typical problem instances derived from sensor network configuration problems.

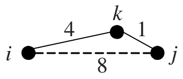
While the above view adopts a static graph model of sensor networks, practical sensor networks are rather dynamic: nodes may fail and the quality of wireless links fluctuates over time. To make our algorithm applicable to such realistic settings, we propose a set of rules to repair a sensor network configuration in case of node failures, additions, and link quality changes. Also, we study the optimality of our algorithm using link quality traces collected from a real sensor network deployment.

2 Preliminaries

We model the multi-hop network subject to configuration as a graph $G = (V, E)$. In our application of the facility location problem, a network node takes on the role of a client and that of a potential facility at the same time, that is, $F = C = V$. In some cases, only a subset nodes have the necessary capabilities (e.g., remaining energy, available sensors, communication bandwidth, or processing power) to execute a service. In such cases, the nodes eligible as facilities can be selected beforehand based on their capabilities [9], which results in $F \subseteq C = V$. When clients are connected to facilities, we will use $\sigma(j)$ to refer to the facility that connects a client j.

Based on the problem at hand, one may choose a particular setting of opening costs f_i and connection costs c_{ij}. In most settings, for a network link $(i, j) \in E$, the respective communication cost c_{ij} will be set to some link metric that can be determined locally at the nodes, e.g., based on dissipated energy or latency. Some approximation algorithms require that the costs c_{ij} constitute a *metric* instance. A *metric* instance requires that, for any three nodes i, j, k, the direct path is shorter than a detour ($c_{ij} \leq c_{ik} + c_{kj}$).

However, if connection costs c_{ij} should represent the transmit power used for sending, these are often proportional to the square of the geographic distance between i and j, which results in non-metric instances, for example:



If the input to a facility location algorithm is non-metric, the problem is particularly hard to solve (see Section 3 below). However, one may obtain a *metric* instance by ignoring non-metric links and setting c_{ij} to the cost of a shortest path between two nodes i and j. In multi-hop networks, the required shortest-paths computation can be achieved using a local flood around the current node.

When addressing settings in which facilities and clients can be an arbitrary number of network hops apart, we will always compute c_{ij} via shortest-paths. We refer to this metric problem setting as *multi-hop*. Alternatively, we will consider a second (constrained) version of the problem, in which we require that every client is connected to a facility which is its direct network neighbor. We denote this constrained problem definition as *one-hop*. One-hop instances are inherently non-metric, as missing links $(i, j) \notin E$, modeled by $c_{ij} = \infty$, violate the metric property.

3 Related Work

An ample amount of literature exists on (centralized) approximation algorithms for the NP-hard facility location problem [10]. Such centralized algorithms are not applicable as these would require a prohibitive communication overhead associated with collecting the whole network topology at a single point (e.g., at the network basestation).

For *non-metric* instances of the facility location problem, even approximations are hard to come by: As the set cover problem can be reduced to (non-metric) facility location, the best achievable approximation ratio (even with a centralized algorithm) is logarithmic[1] in the number of nodes [11]. A classic and simple algorithm [12] already comes close to this lower bound. Distributed approximations are rare: [7] solve non-metric facility location even in a constant number of communication rounds. However, the algorithm requires that a coefficient ρ, which is computed from a global view of the problem instance, is distributed to all nodes before algorithm execution – which prevents it from being used "as-is" in practice. Moreover, the algorithm requires global communication among all relevant clients and facilities and therefore can only efficiently be used in the *one-hop* setting where such communication can be implemented efficiently by wireless broadcast. Finally, the best approximation factor it can obtain, which is independent of the problem instance, is on the order of $O(\log(m+n)\log(mn))$ where m and n denote the number of facilities and clients, respectively.

For *metric* instances of the facility location problem, much better approximation factors $\in O(1)$ can be achieved. While it has been shown [13] that a polynomial-time algorithm cannot obtain an approximation ratio better than 1.463, a centralized algorithm [14] already provides a solution that is at most a factor of 1.52 away from the optimum. For the metric case, to our knowledge only one distributed algorithm has

[1] This holds unless every problem in NP can be solved in $O(n^{O(\log \log n)})$ time.

been mentioned [8] which solves only a constrained version of the problem in which facilities and clients may be at most 3 hops away. It provides a $3 + \epsilon$ approximation factor derived from a parallelized execution of a respective centralized algorithm [15] and is formulated in terms of a synchronous message passing model. The same paper [8] includes additional versions, which restrict the facility location problem in one way or another. Only recently, a highly-constrained version of the facility location problem has been addressed in a distributed manner [6]. Finally, a distributed algorithm based on hill-climbing [16] addresses a version of the problem in which exactly k facilities are opened. However, the worst-case time complexity and the obtained approximation factor are not discussed explicitly.

In this paper, we develop a distributed version of a centralized algorithm [17] which provides an 1.61 approximation factor with metric instances. Compared to related work, our work improves on the approximation factor achievable in a distributed manner. Moreover, we provide the adaptations required to execute this algorithm in multi-hop networks for which, to our knowledge, no efficient algorithm with guaranteed worst-case approximation factor exists. Finally, compared to [7,8], our algorithms do not require a synchronous message passing model. Instead, they perform synchronization among network neighbors implicitly as nodes wait for incoming messages.

In the remainder of the paper, we briefly summarize the centralized approximation algorithms [17] our work is based on in Section 4. We then describe their distributed re-formulation in two steps. The first variant, in Section 5, still requires global communication, namely that all clients communicate with all relevant facilities in each step, and is therefore only applicable to the *one-hop* setting, where this can be efficiently implemented as a wireless broadcast. In the second step, we use this algorithm as a subroutine in the algorithms of Section 6, which distribute messages only to a local neighborhood around the sending node and may therefore be used in multi-hop networks. Finally, we provide experimental results in Section 7 and an outlook to future work in Section 8.

4 Centralized Algorithms

Jain et al. [17] devised two centralized approximation algorithms for the facility location problem. Both use the notion of a *star* (i, B) consisting of a facility i and an arbitrary choice of clients $B \subseteq C$ (in clustering terminology, a star corresponds to a cluster leader and a set of associated slave nodes). The first is shown in Algorithm 1. In its core step (line1.3), the algorithm selects the star (i, B) with best (lowest) cost efficiency. The cost efficiency of a star is defined as

$$c(i, B) = \left(f_i + \sum c_{ij}\right) / |B| \tag{1}$$

and represents the average cost per client which this star adds to the total cost.

Therefore, in each step, the algorithm selects the most cost-efficient star (i, B), opens the respective facility i, connects all clients $j \in B$ to i (sets $\sigma(\mathrm{j}) = i$), and from this point on disregards all (now connected) clients in B. The algorithm terminates once all clients are connected.

Algorithm 1. Centralized 1.861-approximation algorithm [17]

```
1.1 set $U = C$
1.2 while $U \neq \emptyset$ do
1.3     find most cost-efficient star $(i, B)$ with $B \subseteq U$
1.4     open facility $i$ (if not already open)
1.5     set $\sigma(j) = i$ for all $j \in B$
1.6     set $U = U \setminus B$
1.7     set $f_i = 0$
```

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 sorted by ascending connection cost to i. Any most cost-efficient star spanning some $k = |B|$ clients will consist of the first k clients with lowest connection costs – all other subsets of k clients can be disregarded as these cannot be more efficient. Hence, at most $|C|$ different sets must be considered.

When a facility i is 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 been opened before in order to serve other clients. For metric instances, Algorithm 1 provides a 1.861 approximation factor. Note that line 1.7 constitutes the only difference to a classic algorithm [12], whose approximation factor for metric instances is much worse. An even better approximation factor of 1.61 can be obtained when changing the above algorithm to additionally take into account the benefit of opening a facility i for clients that are already connected to some other facility. This involves two changes.

First, this requires that a revised cost-efficiency definition is used in line 1.3. We let $B(i)$ denote the set of clients j which are already connected to some facility $\sigma(j)$ and would benefit if i would be opened as their connection cost to i would be lower than their current connection cost $c_{\sigma(j)j}$, i.e.,

$$B(i) = \left\{ j \in C \text{ with } \sigma(j) \neq \mathit{none} \text{ and } c_{ij} < c_{\sigma(j)j} \right\}. \tag{2}$$

The cost efficiency of a star (i, B) can now be restated as

$$c(i, B) = \left(f_i + \sum_{j \in B} c_{ij} - \sum_{j \in B(i)} (c_{\sigma(j)j} - c_{ij}) \right) / |B|. \tag{3}$$

A second analogous change is made to line 1.5. In addition to the clients which are part of the most-efficient star (i, B), all already-connected clients $B(i)$ which benefit from switching are connected to i. For this, line 1.5 becomes

set $\sigma(j) = i$ for all $j \in B \cup B(i)$.

The authors prove [17] that this change improves the approximation factor to 1.61 for metric instances. In the following, we will present a distributed version of this 1.61-algorithm. In the discussed distributed adaptations, we will always use the revised cost-efficiency definition of Eq. (3).

5 One-Hop Approximation

Consider the distributed algorithms given in Algorithm 2 (for facilities) and 3 (for clients). We will show below that they perform the exact same steps as the centralized Algorithm 1. While these algorithms require that each client communicates with each facility and vice versa, the algorithms can be also applied "locally" such that each node communicates only with its network neighbors. This way, they can be used to compute a solution to the *one-hop* version of the facility location problem, for example, to compute an energy-efficient clustering that takes costs of individual links into account. Unfortunately, this constrained problem version results in a non-metric instance (see Section 2) and thus the approximation guarantee of 1.61 cannot be preserved. However, in the next section, we will use these algorithms as a subroutine to obtain an algorithm that maintains the approximation factor of 1.61 for multi-hop sensor networks. Moreover, we will show that it computes good solutions, nevertheless, in our experimental results of Section 7.

We assume that after an initial neighbor discovery phase, each client j knows the set of neighboring facilities, which it stores in the local variable F_j, and the connection costs c_{ij} to facilities $i \in F_j$. Vice versa, each facility i knows the set of neighboring clients C_i and c_{ij} of all $i \in C_i$. In the following we will simply write C and F, as the respective indices i and j can be deduced from the context.

Algorithm 2. Distributed formulation of Algorithm 1 for Facility i

```
2.1  set $U = C$
2.2  **repeat**
2.3      find most cost-efficient star $(i, B)$ with $B \subseteq U$
2.4      **send** $c(i, B)$ to all $j \in U$
2.5      **receive** "connect-requests" from set $B^* \subseteq U$
2.6      **if** $B^* = B$ **then**
2.7          open facility $i$ (if not already open)
2.8          **send** "open" to all $j \in F$
2.9          set $U = U \setminus B$
2.10         set $f_i = 0$
2.11     **receive** $\sigma(j) \neq none$ from set $C_a$
2.12     set $U = U \setminus C_a$
2.13 **until** $U = \emptyset$
```

As in Algorithm 1, this time each facility i maintains a set U of unconnected clients which is initially equal to C (line 2.1). 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 2.3-2.4). In turn, the clients can expect to receive cost-efficiency numbers $c(i, B)$ from all facilities $i \in F$ (line 3.2). 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 $c(i^*, B)$ with a "connect request" (line 3.4). In turn, facilities collect a set of clients B^* which have sent these "connect requests" (line 2.5). 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 2.6). This is necessary, as it could happen that some clients

Algorithm 3. Distributed formulation of Algorithm 1 for a Client j

```
3.1  repeat
3.2      receive $c(i, B)$ from all $i \in F$
3.3      $i^* = \text{argmin}_{i \in F}\, c(i, B)$ // use node ids to break ties among equal $c(i, B)$
3.4      send "connect-request" to $i^*$
3.5      if received "open" from $i^*$ then
3.6          set $\sigma(j) = i^*$
3.7          send $\sigma(j)$ to all $i \in F$
3.8  until connected
3.9  on "open" from $i$ with $c_{ij} < c_{\sigma(j)j}$
3.10     set $\sigma(j) = i$
3.11     send $\sigma(j)$ to all $i \in F$
```

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 B from U, and sets its opening costs to 0 (lines 2.7-2.10) as in the centralized algorithm.

If a client j receives such an "open" message from the same facility i^* which it had previously selected as the most cost efficient, it can connect to i^* (lines 3.5-3.6). Further, in line 3.7, client j notifies all facilities that it is now connected to i^*, which update their sets of unconnected clients U in lines 2.11-2.12.

Once connected, clients simply switch the facility they are connected to in case a closer facility becomes available (lines 3.9-3.10). This feature enables the 1.61 approximation factor. Note that whenever a client changes its facility $\sigma(j)$, it informs all facilities about this (lines 3.7 and 3.11). All these $\sigma(j)$ messages include the associated connection costs $c_{\sigma(j)j}$ and will be received in line 2.11 of the facility algorithm. By the next iteration, facilities will have received $\sigma(j)$ and $c_{\sigma(j)j}$ from all relevant clients, and will therefore be able to correctly compute the most cost-efficient star (line 2.3) according to Eq.(3).

Discussion. In the following, we argue that the distributed and the centralized versions are equivalent. For this, we denote one execution of the inner loops at Algorithms 3 and 4 as a round. Note that the distributed version opens some stars out-of-order, that is, earlier than the centralized version. The following lemma states that these stars are disjunct from any star that might follow and has lower cost-efficiency.

Lemma 1. *Let U^k be the set of uncovered clients prior to the beginning of round k. If a client j is part of a star (i, B) opened by the distributed algorithm in round k, then there is no star (i', B') considering $B' \subseteq U^k$ with $j \in B'$ and $c(i', B') < c(i, B)$.*

Proof. Assume the contrary, namely that a star (i', B') exists with $c(i', B') < c(i, B)$ and say j is a client in $B' \cap B$. Note that $B' \subseteq U^k$, and therefore i' will choose some star (i', B'') with cost-efficiency $c(i', B'') \leq c(i', B')$ in line. However, as (i, B) is opened in round k, client j has sent its connect request to i and not to i', which implies $c(i', B') \geq c(i, B)$ and contradicts the assumption.

Given the above, we can show that the stars opened by the distributed algorithm can be re-ordered to correspond to the execution of the centralized algorithm.

Theorem 1. *The distributed and centralized versions are equivalent.*

Proof. We sequentialize the distributed algorithm as follows: In the sequentialized version we open only one star (the globally most cost-efficient star) per round. Further, we postpone opening a star (i, B) which has been opened in parallel by the distributed algorithm to a later round prior to which all stars (i', B') with $c(i', B') < c(i, B)$ have been processed. Let (i', B') denote one such star. Because of Lemma 1, $B' \cap B = \emptyset$, and therefore opening (i', B') ahead of time does not remove any client in B from U and therefore does not interfere with opening (i, B). Similarly, postponing any (i, B) will not allow that a more cost-efficient star including elements of B is formed earlier – again by Lemma 1. Postponing (i, B) can further influence (raise) the cost-efficiency of the stars (i', B') as it changes the set $B(i)$ for these facilities and thus may change the order in which these are processed. However, as by Lemma 1 all these stars are mutually disjunct, the order in which they are opened does not affect total costs. Finally, all stars opened in parallel are disjunct and re-ordering them does not change algorithm execution.

Therefore, the sequentialized version opens the same stars as the distributed algorithm. Moreover, as the sequentialized version opens the most cost-efficient star in every round, it implements the execution of the centralized algorithm.

Nevertheless, the worst-case number of rounds required by Algorithms 2 and 3 remains linear in the number of nodes, 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), thus the second cannot open. In this example, only the facility at the end of the chain can be opened in one round. Similarly, once at least one facility is open, it could happen that in each round only one client connects to this facility. The worst-case runtime is therefore $O(n)$, in which n is the number of network nodes.

The linear number of rounds required in the worst-case would constitute a very high overhead in large-scale sensor networks. However, a worst-case configuration on a larger scale is highly improbable (as we will show in Section 7), and the approximation factor inherited from the centralized version is intriguing, particularly because the algorithm performs even much better than 1.61 on average instances. We will evaluate the *average* number of rounds required for typical instances in sensor networks and the optimality gap when the algorithm is executed with such instances in Section 7.

As we mentioned, however, the above algorithm only retains its approximation factor with metric instances, and as any metric instance is essentially a complete graph, it requires global communication between all clients and facilities. This is only efficient in few settings, for example when all nodes hear each other over the wireless broadcast medium. In the next section we use the algorithms of this section as subroutines in an adapted "local" version that functions properly in multi-hop networks.

6 Multi-hop Approximation

The described algorithm can be changed to work in multi-hop settings using only a slight adaptation. As it turns out, if connection costs represent shortest paths between network nodes, the communication performed by the algorithms can be restricted to small network neighborhoods. Specifically, if one is interested in determining whether a facility i has a cost-efficiency of less than a certain threshold s, it is sufficient to consider only clients j that are reachable by i over a path with costs of at most s, i.e., clients j with $c_{ij} \leq s$. To see this, consider the definition of a facility's cost-efficiency and assume that some star's cost efficiency $c(i, B) \leq s$. One can always obtain an even smaller cost-efficiency once one removes the clients $j \in B'$ which have $c_{ij} > s$, that is, $c(i, B \setminus B') < c(i, B)$. Similarly, given a facility i, the clients with $c_{\sigma(j)j} > c_{ij}$ will not occur in the set $B(i)$ of Eq. (3). Therefore, it is sufficient that clients j which are newly connected to $\sigma(j)$ distribute $\sigma(j)$ only to facilities i with cost $c_{ij} < c_{\sigma(j)j}$.

In an outer loop added around Algorithms 2 and 3, we therefore exponentially increase the communication scope s, that is, the maximum distance over which messages are forwarded. Specifically, given a certain scope s, a message is only flooded within a localized neighborhood $N_s(i)$ around the sending node i, where $N_s(i) := \{j \in V \text{ with } c_{ij} \leq s\}$. Note that if the direct link (i, j) is not present in the network graph, c_{ij} representing the shortest path from j to i can be determined on the fly while flooding a message within $N_s(j)$. Nodes simply stop forwarding a message if it has covered a distance of larger than s or if it has already been received over a shorter path.

The updated versions are given in Algorithm 4 (clients) and Algorithm 5 (facilities). In the following, we will respectively use C_s and F_s to refer to client and facility nodes within scope s of the current node.

Algorithm 4. Multi-Hop Adaptation of Algorithm 3 for a Client j

```
4.1  set $s = 1$, set $\sigma(j) =$ none
4.2  repeat
4.3      set $s = s \times a$
4.4      send "start($s$)" to all $i \in F_s$
4.5      if no "begin(s)" received then continue
4.6      repeat
4.7          receive $c(i, B)$ from all facilities $F_s$
4.8          set $F_a = \{i \in F_s$ with $c(i, B) \leq s\}$
4.9          if $F_a \neq \emptyset$ then
4.10             $i^* = \text{argmin}_{i \in F_a} c(i, B)$ // use node ids to break ties
4.11             send "connect-request" to $i^*$
4.12             if received "open($s$)" from $i^*$ then
4.13                 set $\sigma(j) = i^*$
4.14                 send $\sigma(j)$ to all $i \in F_s$
4.15     until connected or $F_a = \emptyset$
4.16 until connected
4.17 on "open($s^*$)" from $i$ with $c_{ij} < c_{\sigma(j)j}$
4.18     set $\sigma(j) = i$
4.19     send $\sigma(j)$ to all $i \in F_{s^*}$
```

Algorithm 5. Multi-Hop Adaptation of Algorithm 2 for Facility i

```
5.1  set $s = 1$
5.2  **repeat**
5.3      set $s = s \times a$
5.4      **if** *"start($s$)" received* **then send** "begin($s$)" to all $j \in C_s$ **else continue**
5.5      **query** $\sigma(j)$ from all $j \in C_s$
5.6      set $U_s = \{j \in C_s \text{ with } \sigma(j) = none\}$
5.7      **repeat**
5.8          find most cost-efficient star $(i, B)$ with $B \subseteq U_s$
5.9          **send** $c(i, B)$ to all $j \in U_s$
5.10         **if** $c(i, B) \leq s$ **then**
5.11             **receive** "connect-requests" from set $B^* \subseteq U_s$.
5.12             **if** $B^* = B$ **then**
5.13                 open facility $i$ (if not already open)
5.14                 **send** "open($s$)" to all $j \in C$
5.15                 set $U_s = U_s \setminus B$, set $f_i = 0$
5.16             **receive** $\sigma(j) \neq none$ from some clients $B' \subseteq U_s$
5.17             set $U_s = U_s \setminus B'$
5.18     **until** $U_s = \emptyset$ *or* $c(i, B) > s$
5.19 **until** $s > s_{\max}$
```

In the outer loop, the considered scope s is raised exponentially (lines 4.3 and 5.3). To initialize an outer round, clients, which have not yet been connected, send a "start" message containing their current scope s to all facilities in scope (line 4.4). In turn, facilities wait for at least one such "start" message for a certain time (line 4.5) upon which they reply "begin(s)". The waiting period must be long enough to allow relevant clients to send the respective start messages and finish earlier rounds. If no "start" messages were received, facilities simply advance to the next outer round (line 5.4) to wait for "start" messages from a larger scope. Clients, analogously, wait and then skip the current round if no neighboring facility has sent "begin".

A start message sent by a client j thus triggers execution of one outer round at all the facilities in scope F_s. Facilities then query all clients in scope for their status $\sigma(j)$ in line 5.5 and compute the set of yet unconnected clients U_s. This query-reply cycle allows the facility to wait for all relevant clients to catch up to the current scope s. Clients reply to this query once they have reached scope s – note that we have omitted the respective code in the client algorithm. Similarly clients can wait for facilities lagging behind in line 4.7 where they expect to receive a message from all facilities in scope.

After this initialization, facilities execute Algorithm 2 in an inner loop (lines 5.7-5.18) and clients react accordingly (lines 4.6-4.15) implementing Algorithm 3. Compared to Algorithms 2 and 3 the termination conditions of the inner loops must be changed to allow clients and facilities to proceed to a larger scope in a properly synchronized manner. As with the 1-hop version, clients terminate their inner loop once they are connected (line 4.15) and facilities once no active clients remain in scope (line 5.18). In addition, within an inner-loop with scope s, the algorithm should only consider stars (i, B) with cost-efficiency $c(i, B) < s$. Therefore, facilities only proceed with the current inner loop as long as they are efficient enough for this scope (lines 5.10 and 5.18) while in turn clients only proceed with their inner loop as long as there is a facility in scope that is efficient enough to connect them (lines 4.8,4.9 and 4.15).

Finally, once a client has been connected (4.17-4.19), it acts analogously to Algorithm 3: It simply changes its facility if this is beneficial and notifies all relevant facilities about it. Here the client can synchronize to the scope s^* of the sending facility as it is included in the received "open" message to ensure that all relevant facilities are informed. Note that the messages sent in line 4.19 are also received by facilities still performing their inner loop in line 5.16.

Discussion. The algorithms presented in this section enhance Algorithms 3 and 4 by making them "local", meaning that they do not need to communicate with all relevant facilities but only to the ones within a confined neighborhood. This allows to perform shortest-paths computations in these confined neighborhoods which, in turn, give rise to metric instances and preserve the approximation factor of Algorithm 1.

An additional outer loop provides for both, an adequate expansion of the involved communication scope and for sufficient synchronization of the nodes in scope without depending on a synchronized communication model. Because clients and facilities may repeatedly have to wait in lines 4.5 and 5.4, respectively, the worst-case runtime becomes $O(n \log_a s_{\max})$ where $s_{\max}$ denotes the cost efficiency of the least efficient star which occurs in the network and n denotes the total number of participating nodes. However, the maximum number of rounds involving actual communication is smaller. If no unconnected clients or eligible facilities are present, the involved nodes do not communicate in their inner loop at all. Instead, they simply skip the inner loop. In turn, in rounds involving communication, a client or facility can be a single point of activity only once during algorithm execution. Therefore, the number of required communication rounds is still in $O(n)$.

Dynamic Re-configuration. In real-world deployments of sensor networks, link qualities change over time and nodes may fail. To accommodate for *major* changes in the network topology, the algorithms are re-executed at regular intervals. As such re-starts involve relatively high overhead, these are performed only infrequently (e.g., once a day). In between such re-starts, a client j combines periodic re-evaluations of link costs c_{ij} (within a local scope of size $c_{\sigma(j)j}$) with a liveness check on the facility $\sigma(j)$. In both cases, if $\sigma(j)$ has failed or a closer open facility has been found, client j re-connects to the closest open facility. In Section 7, we will show that such adaptations suffice to maintain a close-to-optimal configuration over longer periods of time.

7 Experimental Results

In the following, we show results from two distinct sets of experiments. The first, detailed in Section 7.1, is based on simulations which test the scalability of the proposed algorithms. The second, detailed in Section 7.2 tests the applicability of the proposed algorithms to operational networks with dynamic links.

7.1 Scalability

In the experiments based on simulations, we uniformly deployed a variable number of nodes (x-axis) onto a 300m by 300m area. The network graph has an edge $(i, j) \in E$ if the nodes i and j are less than $30m$ apart (this number stems from a model that is based on the characteristics of the CC1000 transceiver used on BTnodes [18] and Berkeley Motes). Assuming that nodes can control their transmit power, for $(i, j) \in E$, we set connection costs $c_{ij} \sim g(i, j)^2$ where $g(i, j)$ denotes the distance in meters between i and j and normalize them such that $c_{ij} \in [0, 1]$.

Scenarios. To test our algorithms with a range of applications, we examined three different parameterizations of the facility location problem, of which qualitative results are shown in Figure 1. In the first, we set opening costs $f_i = 1$ and additionally require that clients and facilities must be neighbors. We show a solution obtained by the one-hop Algorithms 2 and 3 on such an instance in Figure 1(a).

Further, we tested the multi-hop Algorithms 4 and 5 in two different settings. In the first, we set $f_i = 5$ to denote that a high effort is required to operate a cluster leader, of which an example result is shown in Figure 1(b). In the second scenario, shown in Figure 1(c), we assumed that cluster leaders must send much data to the network basestation and therefore their operation costs increase with their network distance to the sink (yielding smaller stars close to the sink and larger ones further away).

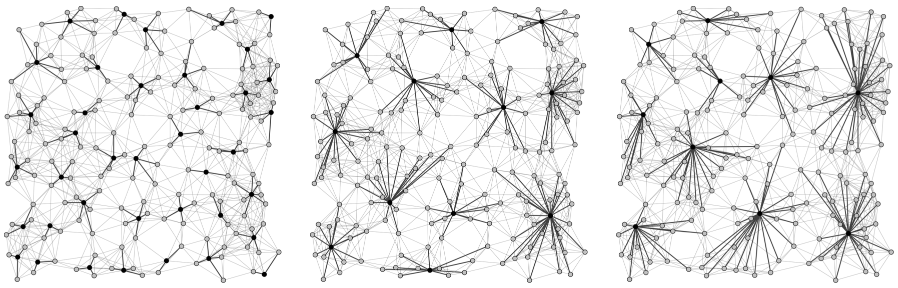

(a) $f_i = 1$ (b) $f_i = 5$ (c) $f_i = 2 \times D(\text{sink}, i)$

Fig. 1. Effects of varying opening costs ($D(\textit{sink}, i)$ denotes the shortest-path distance to the sink, which is located in the upper left corner of the simulated area)

One-Hop Clusters. In the *one-hop* setting (Figure 1(a)), we evaluated the costs of configurations produced by different algorithms while varying the number of nodes in the simulation area (that is, the node density). The results are given in Figure 2(a) which shows the costs obtained with the following five methods.

One-hop denotes the simple one-hop algorithms of Section 5. Respectively, *one-hop IP* refers to the optimal configuration of the constrained case which requires clients to connect to facilities which are direct network neighbors. Further, *multi-hop* denotes the multi-hop algorithm described in Section 6, which has a 1.61 approximation guarantee. Here, clients may connect to facilities which are an arbitrary number of hops away. Respectively, *multi-hop-IP* computes the optimal solution to the facility location problem, in which facilities and clients may be multiple hops apart and the instance is made metric by a centralized shortest-paths computation. Finally, *MDS-IP* denotes the optimal solution to the minimum dominating set problem, in which dominator nodes represent open facilities and slave nodes are clients that connect to the closest dominator node. The costs are computed using the original (non-metric) instance.

The costs of a minimum dominating set (*MDS-IP*) which suffer from expensive long links mark one end of the optimization spectrum. Here we argued that facility location can provide a more energy efficient configuration. On the other hand, the optimal facility-location based configuration (*multihop-IP*) marks the other end as it represents a lower bound for the employed approximation algorithms.

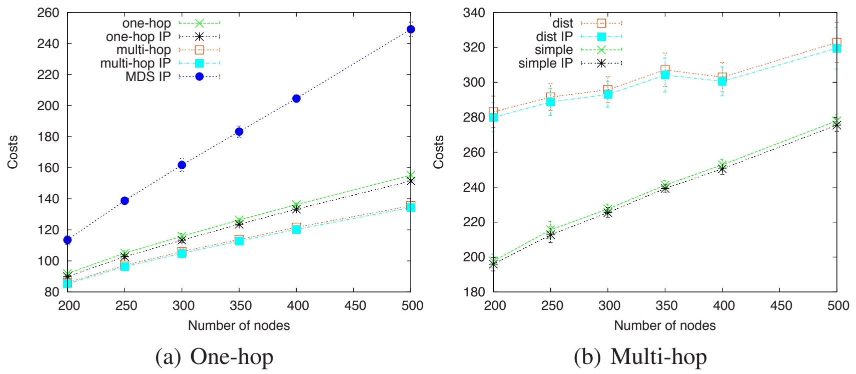


Fig. 2. Performance of one-hop and multi-hop algorithms

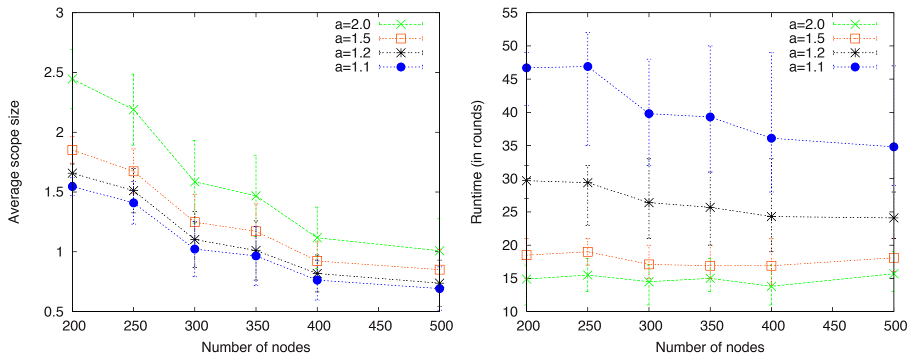
The *one-hop* algorithm performs well and is even close to the respective optimal configuration *one-hop IP*, although it operates on a non-metric instance and thus without a guaranteed approximation factor.

Note that in this particular setting, the constrained versions, which require facilities and clients to be direct neighbors (*one-hop* and the optimal *one-hop IP*), are not far away from the *multi-hop* results and the optimum of the unconstrained case (*multi-hop IP*). This is due to the low opening costs we used, which are set to $f_i = 1$ for all facilities. With larger opening costs, multi-hop solutions would benefit more from larger stars.

Multi-Hop Clusters. In the experiments shown in Figure 2(b), we additionally evaluate the quality of the solutions obtained by the multi-hop algorithm with the two different opening cost settings shown in Figures 1(b) and 1(c). In the first (denoted as *simple*) we set opening costs to a constant $f_i = 5$ which corresponds to configurations as shown in Figure 1(b). In the second, denoted as *dist*, we apply the heuristic shown in Figure 1(c), where the opening costs correspond to twice the costs of the shortest path to the sink. In both cases, the results of the distributed implementation are very close to the achievable optimum computed by CPLEX on the same instance.

Runtime and Overhead. In the experiments shown in Figure 2(b), the scope s started out with 0.2 and a was set to 2, thus doubling the scope in each outer round. Note, however, that these two parameters do not influence the quality of the obtained solution. Rather, they determine the trade-off achieved between the runtime of the algorithms and the scope within which messages are sent. On the one hand, the smaller a is set, the more one may be sure that scopes are not increased too far (in vain). On the other hand, the required number of outer rounds until termination increases with lower a-values.

Figure 3 demonstrates this trade-off as observed in the simulation run corresponding to Figure 2(b). In Figure 3(a) we show the average scope with which messages were sent during algorithm execution, given different settings of a (the scope s always starts at 0.2). The lower we set a, the better the results as the scope is increased by smaller amounts. Note that in general, the effort involved in the execution of our algorithm is proportional to the "locality" implied by the problem instance: On the one hand, if opening costs are high (here $f_i = 5$), a facility will generally connect clients in a larger neighborhood (as seen in Figure 1(b)). On the other hand, the experienced scopes are even much lower with small opening costs (e.g., for $f_i = 1$, not shown).



(a) Scope (b) Rounds

Fig. 3. Average scope size vs. total runtime (in rounds). In Figure 3(b) the error bars denote the maximum and the minimum that occurred.

In contrast, in Figure 3(b), we show the runtime in rounds (one round corresponds to one execution of the inner loop) of the multi-hop algorithm on the same instances. Note that, while previously the error bars indicated confidence intervals of 95%, we use them in Figure 3(b) to mark the maximum and minimum values that occurred in 10 random instances (as we are particularly interested in the maximum value). The results show that – while in theory the worst case runtime can be large – in typical instances based on multi-hop networks the runtime is sufficiently small and does not even grow with the number of nodes. Moreover, based on the trade-off between runtime and scope size, the runtime improves with higher a values. Finally, the scope size decreases with increasing network density. This is due to the fact that, given certain opening costs, the algorithms will connect stars of around the same size (namely, facilities are opened once enough clients are connected to pay for opening them). Therefore, smaller stars are opened in denser networks and the cumulated communication overhead stays the same.

7.2 Network Dynamics

One open question is whether such, albeit close-to-optimal solutions, can provide a benefit for real-world deployments in which the network topology changes over time. To obtain realistic link qualities, we extended a testbed of 13 TMote Sky modules that gather temperature, humidity, and light measurements from our office premises to record network topology information as well. Next to its sensor measurements, every 5 seconds, a node reports the set of nodes from which an application-layer message has been received since the last update.

Such topology information received from each node i allows to compute a (packet-level) link quality estimate $e_{ij}(t)$ for each network link directed from j to i [19]. The estimate $e_{ij}(t)$ is based on the packet success rate $r_{ij} = \frac{\text{packets received in T}}{\text{packets expected in T}}$ which is smoothened using an exponentially weighted moving average such that $e_{ij}(t) = \alpha r_{ij}(t) + (1-\alpha)e_{ij}(t-1)$. In our experiments, we set α=0.6 according to [19] and T to 300 s. We transform the quality estimates $e_{ij} \in [0,1]$ into link cost estimates by setting $c_{ij} = 1+10(1-e_{ij})$ if $e_{ij} > 0.5$ and $c_{ij} = \infty$, otherwise. Further, we set opening costs to constant $f_i = 2$.

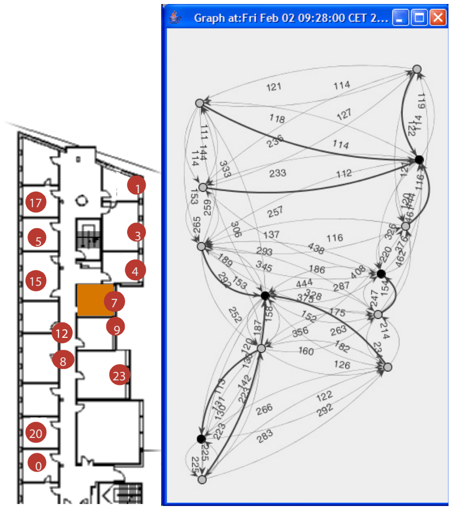


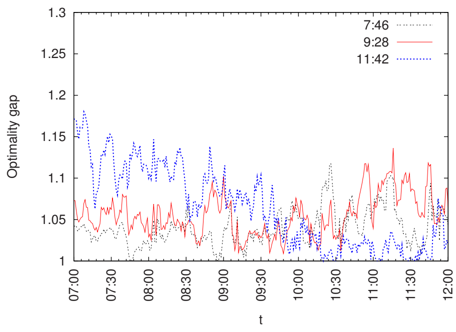
Fig. 4. Deployment plan (left); network topology at 9:28 a.m. showing $c_{ij} \times 100$ and computed configuration (right)

To give the reader an impression of the examined networks, Figure 4 shows our mote deployment, the resulting network topology, and a configuration computed by the multi-hop algorithms.

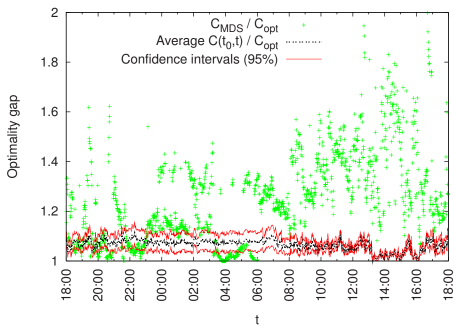
Given the link costs $\{c_{ij}(t_0)\}$ observed at a certain time of the experiment t_0, we let the presented multi-hop algorithms compute a configuration (a set of open facilities and assigned clients), whose costs $C(t_0, t)$ vary with t as link qualities change over time. Once a configuration has been computed, only small dynamic adaptations (detailed in Section 6) are performed.

In Figure 5(a), we show the ratio between $C(t_0, t)$ and the costs of an optimal configuration C_{opt} computed by CPLEX – for configurations computed at three arbitrarily chosen instants of time t_0. Observe how at $t = t_0$, e.g. at 7:46 or at 11:42, the respective optimality gap is close to 1. As expected, however, this is not always the case. For example the configuration obtained at t_0=9:28 is not optimal even at this time.

In Figure 5(a), one can observe how the time t_0 at which the initial configuration is computed influences the respective outcome of $C(t_0, t)$. To obtain more general results, t_0 is randomly drawn from the total 24 hour interval corresponding to available topology data and used to compute the respective curve $C(t_0, t)$ in 20 repeated simulation runs. The ratio of the average $C(t_0, t)$ to the costs of the optimal configuration is shown Figure 5(b). In addition, Figure 5(b) shows the costs C_{MDS} of a minimum dominating set computed by CPLEX for each instant of experiment time. The latter costs can be



(a) $C(t_0, t)/C_{\mathrm{opt}}$ for different t_0 during 5 hours



(b) Average $C(t_0, t)/C_{\mathrm{opt}}$ vs. MDS during 24 hours

Fig. 5. Solutions' optimality over time

used as an assessment of whether a much faster MDS approximation, which can be re-executed frequently, could out-perform a facility location algorithm executed more rarely. As said earlier, however, MDS-based configurations require slaves to use expensive links (with poor link quality estimates) to communicate with their cluster leader. Such "bad" links are often the most volatile and cause the costs of an MDS-based configuration to diverge significantly from an optimal configuration. While this is not always the case (Figure 5(b) has portions in which MDS is close-to-optimal), one can observe that facility-location based configurations, which focus on high-quality links, are robust with respect to varying link qualities. The observed gap to an optimal configuration remains small – in the observed 24 hours it stayed below 10% at all times.

8 Conclusion and Outlook

In this paper, we motivated the use of facility location algorithms to address configuration tasks in multi-hop networks as they can flexibly implement many sensor-network configuration problems, such as an energy-efficient clustering, a clustering in which cluster leaders can connect nodes through multiple hops, or a configuration in which cluster leaders are chosen based on their distance to the sink. We claim that many more such applications of the problem can be found.

Further, we have shown that algorithms which are very good in theory (with an approximation factor of 1.61 whilst the theoretically best polynomial algorithm cannot be better than 1.463) can be feasibly transformed for distributed execution. The transformations we described resulted in (to our knowledge) the first facility location algorithm which can be efficiently executed in multi-hop networks.

In the experimental evaluation, we were able to show that although our algorithm exhibits a linear worst-case runtime, in typical sensor-network instances it terminates in only few communication rounds. Moreover, by analyzing the scopes within which messages were forwarded during algorithm execution, we showed that the devised algorithm, although equivalent to its centralized ancestor, requires only very local communication. Further, we showed that the distributed algorithm always performs close to the optimal solution, a quality which it inherits from the centralized version [17].

Finally, there is much left to do. The algorithms we described could be made faster, possibly employing a technique inspired by [7], in which stars are connected "fractionally" in small parallel steps and the obtained fractional solution is rounded later. Moreover, in wireless multi-hop networks two "harder" versions of the facility location problem have particular applicability, for which, to our knowledge, no distributed algorithms exist at all: The *capacitated* version, in which a facility can only serve a limited number of clients and the *robust* version, in which every client is connected by k facilities.

Acknowledgments. We would like to thank Thomas Moscibroda for many valuable discussions, in particular on [7,8,17]. This work was supported by NCCR-MICS, a center funded by the Swiss National Science Foundation.

References

1. Karl, H., Willig, A.: Protocols and Architectures for Wireless Sensor Networks. Wiley, Chichester (2005)
2. Cerpa, A., Estrin, D.: ASCENT: Adaptive Self-Configuring Sensor Networks Topologies. In: Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02), New York, NY, USA (June 2002)
3. Basagni, S., Mastrogiovanni, M., Petrioli, C.: A performance comparison of protocols for clustering and backbone formation in large scale ad hoc networks. In: Proceedings of the 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'04) (2004)
4. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. In: Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI'02), Boston, MA, USA (December 2002)
5. Gnawali, O., Greenstein, B., Jang, K.Y., Joki, A., Paek, J., Vieira, M., Estrin, D., Govindan, R., Kohler, E.: The TENET architecture for tiered sensor networks. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SENSYS'06), Boulder, CO, USA (November 2006)
6. Gehweiler, J., Lammersen, C., Sohler, C.: 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), Cambridge, MA, USA (2006)
7. Moscibroda, T., Wattenhofer, R.: Facility location: Distributed approximation. In: Proceedings of the 24th ACM Symposium on Principles of Distributed Computing (PODC'05), pp. 108–117 (2005)
8. Chudak, F., Erlebach, T., Panconesi, A., Sozio, M.: Primal-dual distributed algorithms for covering and facility location problems. Unpublished Manuscript (2005)
9. Frank, C., Römer, K.: 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)
10. Vygen, J.: Approximation algorithms for facility location problems. Technical Report 05950-OR, Research Institute for Discrete Mathematics, University of Bonn (2005)
11. Feige, U.: A threshold of ln n for approximating set cover. Journal of the ACM 45(4) (1998)
12. Hochbaum, D.S.: Heuristics for the fixed cost median problem. Mathematical Programming 22(1), 148–162 (1982)

13. Guha, S., Khuller, S.: Greedy strikes back: Improved facility location algorithms. Journal of Algorithms 31, 228–248 (1999)
14. Mahdian, M., Ye, Y., Zhang, J.: Improved approximation algorithms for metric facility location problems. In: Jansen, K., Leonardi, S., Vazirani, V.V. (eds.) APPROX 2002. LNCS, vol. 2462, Springer, Heidelberg (2002)
15. Jain, K., Vazirani, V.V.: 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), pp. 2–13 (October 1999)
16. Krivitski, D., Schuster, A., Wolff, R.: A local facility location algorithm for sensor networks. In: Prasanna, V.K., Iyengar, S., Spirakis, P.G., Welsh, M. (eds.) DCOSS 2005. LNCS, vol. 3560, Springer, Heidelberg (2005)
17. Jain, K., Mahdian, M., Markakis, E., Saberi, A., Vazirani, V.V.: Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. Journal of the ACM 50, 795–824 (2003)
18. BTnodes (2006), `www.btnode.ethz.ch`
19. Woo, A., Tong, T., Culler, D.: Taming the underlying challenges of reliable multihop routing in sensor networks. In: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SENSYS'03), Los Angeles, CA, USA (November 2003)