A local, distributed constant-factor approximation algorithm for the dynamic facility location problem

Bastian Degener, Barbara Kempkes and Peter Pietrzyk
Heinz Nixdorf Institute
Computer Science Department, University of Paderborn
33095 Paderborn, Germany
degener@uni-paderborn.de, barbaras@uni-paderborn.de, toon@uni-paderborn.de

***Abstract*—We present a distributed, local solution to the dynamic facility location problem in general metrics, where each node is able to act as a facility or a client. To decide which role it should take, each node keeps up a simple invariant in its local neighborhood. This guarantees a global constant factor approximation when the invariant is satisfied at all nodes. Due to the changing distances between nodes, invariants can be violated. We show that restoring the invariants is bounded to a constant neighborhood, takes logarithmic (in the number of nodes) asynchronous rounds and affects each node at most twice per violation.**

***Keywords*-facility location; dynamic; constant factor approximation; local;**

I. Introduction and problem definition

As one of the most studied optimization problems in operation research, the facility location problem captures a large variety of application scenarios. In its original version we are presented with a set of possible warehouse locations and a set of customer locations. Our objective is to decide on which of these possible warehouse locations we want to actually build warehouses. Since maintaining a warehouse incurs high costs, we want to build as few as possible. On the other hand, every customer prefers to be located as close to a warehouse as possible, since costs rise with the distance to the nearest warehouse. This means that we are looking for a placement of the warehouses that minimizes the sum of the costs caused by the customers and the warehouses. Since this problem is NP-complete, approximations of the optimal solution are of interest.

A. Problem definition

In this paper, we will consider a special case of the metric facility location problem defined in the following way: We are given a metric space $M(V, d_t)$ in which V represents a set of nodes and the function $d_t : V \times V \to \mathbb{R}_{\geq 0}$ represents the distance between two nodes $v_i, v_j \in V$ at time $t \in \mathbb{R}_{\geq 0}$. We introduce dynamics by allowing the distances between nodes to change over time. Also, to each node v_i two constant values $f_i, c_i \in \mathbb{R}_{\geq 0}$ (which do not depend on the number of nodes) are provided as input. Every node can fulfill one of two roles: It can either be a *facility* or a *client*. The role of a node is not fixed and can change over time. This means that, at any given time, we can subdivide the set V into the sets F_t and C_t, where F_t contains all the nodes with the facility role at time t, while C_t contains all the client nodes. If the node v_i is a client its demands are denoted by c_i and it causes costs $c_i \cdot d_t^{min}(v_i, F)$, where $d_t^{min}(v_i, F) := min_{v_k \in F}\{d_t(v_i, v_k)\}$ is the distance between v_i and the closest facility to v_i at time t. Otherwise the node v_i is a facility and causes a cost of f_i. The goal is now to assign one of the two roles to each node in such a way that the value $cost_t := \sum_{v_i \in F_t} f_i + \sum_{v_j \in C_t} c_j \cdot d_t^{min}(v_j, F_t)$ is minimized.

We want to provide the nodes with an approximation algorithm that enables them to switch their roles, so that the subdivision into the sets F_t and C_t induced by their roles keeps the value $cost_t$ within a constant factor of the optimal solution. Once, due to changes in the metric function, the current solution ceases to be a constant factor approximation, the goal of the algorithm is to compute a new approximation as fast as possible. Also, as few nodes as possible should be involved in this computation.

The following naming conventions will be used to facilitate talking about the problem we just described: A node acting as a facility will be referred to as *open*, while one acting as client will be called *closed*. Analogously, we say that a node *opens* when it changes its role from client to facility, respectively *closes* when it changes its role from facility to client. When appropriate, we will omit the index t representing the time and refer to d_t, F_t and C_t as d, F and C.

B. Motivation

A possible application for the special case of the facility location problem presented in this paper can be found in the domain of wireless sensor networks. Here it is often crucial to save energy in order to maximize the lifetime of the network. Often these networks are clustered and each cluster contains a node referred to as cluster-head that has

Partially supported by the EU within FP7-ICT-2007-1 under contract no. 215270 (FRONTS) and DFG-project "Smart Teams" within the SPP 1183 "Organic Computing" and International Graduate School Dynamic Intelligent Systems

to fulfill a special, possibly highly energy consuming task, while the other nodes operate in an energy-saving mode, but their energy consumption depends on their distance to the closest cluster-head. Furthermore, we envision the sensor nodes to be deployed to a dynamic environment, e.g. an ocean with currents. This scenario obviously maps exactly to the described facility location problem with the cluster-heads being facilities, the other nodes being clients and the cost (f_i respectively c_i) being energy. The distances between the nodes in the Euclidean space can be modeled by the function d.

Since the problem is not only defined for the Euclidean but for any metric, other applications are possible. A connected, weighted graph, where the edges represent hard-wired connections between nodes and the weights the quality of the connections (i.e. latencies), can be used to represent the underlay for a computer network. This underlay induces a metric on the nodes by defining the distance function d as the shortest path (regarding the sum of weights on the path) in the underlay between two nodes. These latencies change over time and thus dynamics is introduced. As in the example of sensor networks above, finding a clustering minimizing the costs can be an objective here.

C. Our contribution

We introduce a simple distributed algorithm that is executed by each node and is used to determine whether the node should act as a facility or a client in the current situation. Since the distances between the nodes change over time, the algorithm constantly reevaluates its decision and, if necessary, changes the node's role in order to reestablish the approximation. Taken as a whole, the decisions of all the nodes yield a constant factor approximation of the facility location problem.

An important property of our algorithm is the fact that each node only requires local information to be able to execute it: For each node v_i the value r_i, referred to as radius, is computed. In order to compute its radius, v_i requires only information about nodes that are within constant distance of v_i (i.e. a distance independent of the total number of nodes). In addition to the radius, v_i requires information about the current role of all nodes v_j with $d(v_i, v_j)$ bounded from above by a constant. The radius, the c_j's and the current roles of v_i's neighbors v_j is all that is necessary for v_i to determine its own role.

We describe and analyze this algorithm showing that, although the decisions of the nodes are based on local information, the system stabilizes in a solution that yields a global $\mathcal{O}(1)$- approximation. Furthermore, we prove that the process of finding the approximation only requires $\mathcal{O}(\log n)$ communication rounds, and that changes in the role or the radius of a node v_i only affect nodes within constant distance of v_i.

Concerning the difficulty of the problem, we show that our locality constraints can not be tighten without increasing the approximation factor and that it is not possible to bound the distance between v_i and the nodes which are affected by v_i's change of role without using approximation. This proves our results concerning locality to be tight. With respect to the number of required rounds our algorithm can compete with state of the art constant-factor approximation for the non-uniform metric facility location problem algorithms, since so far all these algorithms require $\Omega(\log n)$ rounds. Finding lower bounds in this model is still an important open problem according to [1].

D. Related work

Recently, a 7-approximation for the facility location problem on a complete bipartite graph in a distributed setting was introduced in [2] using a linear programming approach. Different to our approach they do not focus on locality issues, but rather on the size of transmitted messages, limiting them to $\mathcal{O}(\log n)$ bits. A uniform variant of the problem with $f_i = d_i = 1$ for all nodes was also considered in distributed and static settings: In [1], a $\mathcal{O}(k(mn)^{1/k}\log(m+n))$-approximation in $\mathcal{O}(k)$ communication rounds is achieved, with m and n being the number of facilities and clients. In [3], three communication rounds are needed to calculate a $\mathcal{O}(1)$-approximation in the uniform setting. For both algorithms, the number of bits transferred each round is bounded by $\mathcal{O}(\log n)$ and both algorithms cannot be transferred easily to a dynamic setting. The same applies to the non-uniform variant introduced in [1], where the results are harder to compare to ours, since they consider the more difficult non-metric facility location problem. In contrast to our constant factor approximation, they present an approximation factor depending on m and n.

The facility location problem is closely related to clustering problems. For dynamic environments various global event-based algorithms solving clustering related problems are known: a constant factor approximation for the minimal number of centers to cover all points with a given radius [4], solutions to k-center problems with $k = 1$ [5] and even for the facility location problem [6]. In [7] a solution for the dynamic k-center problem that does not require any updates is proposed. However, the size of the required set is augmented to achieve a constant factor approximation.

Some papers use different notions of locality. [8] considers the hop-distance and [9] presents a distributed algorithm for the static facility location problem in unit disk graphs. Contrary to our work, global coordinates are required to achieve a constant factor approximation.

While we present a rigorous analysis, there are also heuristic approaches, e.g. [10]. For a recent survey on placing facilities in wireless mobile networks see [11].

[3] applies the approach of Mettu and Plaxton [12], which is also a major building block in this work. This approach

has been successful in a lot of other settings as well: In the kinetic setting [6], in game theoretic settings [13], and for algorithms working in sub-linear time [14]. However, to our best knowledge the metric facility location problem has never been considered in a dynamic and local setting.

E. The communication model

In order to gather the information required to execute the algorithm, nodes need to communicate with each other. Depending on the considered scenario, different means of communication are applicable.

For the sensor networks scenario we envision indirect communication only. Here, each node constantly provides information about its role and radius to other nodes in its vicinity. This is possible using an idea inspired by the s-bots [15]. Those s-bots are robots that have 24 colored LEDs to represent their state (we assume that one can equip them with a larger number of 'communication bits'). Furthermore, they are equipped with cameras enabling them to acquire the information provided by other robots. Such a communication method could be adopted by nodes in sensor networks and is sometimes referred to as local broadcast.

A possible way of dealing with communication in the computer network scenario is to broadcast the information about the radius and role using the underlay. Since each node is only interested in the nodes within constant distance k, we can limit the messages' traveling distance by adding a label to each message telling how far it is supposed to be sent. In order for this approach to be reasonable, we need to assume that the rate at which the latencies are changing is low and that communication overhead produced by our algorithm is insignificant compared to the huge data streams sent through the network, whose impact on the network is modeled by the distance function d (i.e. the latencies between the nodes).

F. The round model

The execution of the algorithm is embedded in an asynchronous round model. Concerning this model a node's state changes as follows: Starting in an inactive state, the node's state changes to active after an arbitrary amount of time. Now, it determines whether it should change its role and acts according to this decision. After the node's role has been updated, it returns to the inactive state. This sequence of state changes is infinitely repeated by every node.

A round ends after every single node has been active at least once since the end of the previous round. We assume that nodes spend most of the time in the inactive state and that at any point of time at most one node is in the active state.

As we need to define a relationship between the dynamics concerning the distance function d and the nodes' change of states, we introduce the term *event*. An event symbolizes a change in the distance function upon which our algorithm reacts and is defined formally later on. As the time between an event and the point in time until which all nodes became active at least once cannot be bounded, we assume that the activation frequency is high enough compared to the occurrence of events. More precisely, we suppose that at least $c \log n$ rounds take place between two events, where c is a large enough constant. This assumption allows us to guarantee that only a very limited number of nodes, which all are within a constant distance to the event, change their role in order to restore the approximation. Dropping this assumption only allows us to guarantee a constant factor approximation after at least $c \log n$ rounds passed without the occurrence of an event.

G. Modeling dynamics

We allow the values of the distance function d to change arbitrarily over time. The only restrictions we place on the changes is that we require them to happen in a continuous way. Consider the time interval that starts at t_0 and ends at t_1. During this time the distances between the nodes will be changing continuously and thus, due to the execution of our algorithm, events will occur. In order to be able to bound the number of these events, we introduce a characterization, the *dynamics parameter* x ($x \in \mathbb{N}$), of the distance function's changes. For each of the $\binom{n}{2}$ pairs of nodes $\{v_i, v_j\}$, we consider the value $d(v_i, v_j)$ as it changes over time. We say that at time t the pair of nodes $\{v_i, v_j\}$ is in *increasing* mode, if $d(v_i, v_j)$ is increasing, and in *decreasing* mode, if $d(v_i, v_j)$ is decreasing. If $d(v_i, v_j)$ is constant, we say that $\{v_i, v_j\}$ is in *constant* mode. We assume that all node pairs are in constant mode at the beginning. The sum of all changes in the modes over all pairs of nodes during the time interval $[t_0, t_1]$ yields the value for the dynamics parameter x.

Obviously, x can be computed for any kind of continuous changes in the distance function d. If for example the set of nodes represents n sensors moving on trajectories described by polynomials with a degree bounded by a constant and d models the distance between each pair of sensors, then the movement parameter x is bounded by $\mathcal{O}(n^2)$. Another example is the way-point model, where $x \leq m \cdot n^2$ (with m upper bounding the number of points each node travels to in the considered time interval). When trying to upper bound the number of occurring events, it is not important how exactly the distances between nodes change. It is only relevant how often the distances change from increasing to decreasing or vice versa. (If we consider two sensors, it is irrelevant in which direction and speed the sensors move away from each other. All that is needed to bound the number of events is the fact that the distance between them is increasing.) This makes the dynamics parameter x a useful abstraction for all continuous dynamic changes in the distance function d.

II. Description of the algorithm

In the first section, we have already mentioned the radius and events. We now formally introduce these concepts, define an invariant and give a description of our algorithm.

A. Radius of a node

We define the ball $B(v_i, r)$ around v_i with radius r as the set of nodes v_j for which $d(v_i, v_j) \leq r$. The weight of a ball is defined as *weight*$(B(v_i, r)) := \sum_{v_j \in B(v_i,r)} c_j$. A central property of our algorithm is the use of a value called *radius* for each node v_i that can change over time. It was introduced in [12] for a static setting. There, the radius of a node v_i is the value r_i^* satisfying the following equation:

$$\sum_{v_j \in B(v_i, r_i^*)} c_j \cdot (r_i^* - d(v_i, v_j)) = f_i.$$

Observe that the sum on the left side of the equation is continuous and strictly monotonically increasing with r_i^*. Hence, there exists a unique value r_i^* satisfying the equation. Moreover, for any node $v_i \in V$, the radius r_i^* ranges from $\frac{\min_{v_j \in V} f_j}{n \cdot \max_{v_j \in V} c_j}$ to $\frac{\max_{v_j \in V} f_j}{\min_{v_j \in V} c_j}$ [6].

Instead of this exact radius, we use an approximation for it which was first introduced in [6]. The idea is to round the radius to a value which is a power of 2. Because of the bounds for the original radius r_i^*, this leads to $\mathcal{O}(\log n)$ possible values for our radius.

We restate Lemma 2 from [6], because it is crucial for our approximation factor:

Lemma 1. *Let k_1 be the minimum integer k where $\lceil \log_2(\frac{\min_{v_j} f_j}{n \cdot \max_{v_j} c_j}) \rceil \leq k \leq \lfloor \log_2(\frac{\max_{v_j} f_j}{\min_{v_j} c_j}) \rfloor$, such that weight$(B(v_i, 2^k)) \geq f_i \cdot 2^{-k}$. Then $\frac{1}{2} \cdot r_i^* \leq 2^{k_1} \leq 2 \cdot r_i^*$ holds.*

In this paper, we define $r_i := 2^{k_1+1}$ as the radius of node v_i and $k_0 := k_1 + 1$. Because of Lemma 1, $r_i^* \leq r_i \leq 4r_i^*$. Like the original radius r_i^*, r_i can change over time. This can happen when a node moves into distance $\frac{1}{2}2^{k_1} = \frac{1}{4}2^{k_0} = \frac{1}{4}r_i$ of another node v_j (possibly halving r_i) or out of $2^{k_1} = \frac{1}{2}2^{k_0} = \frac{1}{2}r_i$ of another node v_j (possibly doubling r_i), since this is when the weight of the respective ball increases or decreases. Furthermore, $\frac{f_i}{c_i}$ is an upper bound for the original radius r_i^* of node v_i, which is reached when no other node is in v_i's radius. With Lemma 1, this leads to an upper bound of $2^{\lfloor \log(\frac{f_i}{c_i}) \rfloor + 2} \leq 4\frac{f_i}{c_i}$ for r_i. Since f_i and c_i are known to node v_i, it can compute this upper bound. Moreover, since we assume that all f_i and c_i are constant, the value of a radius is bounded from above by a constant. No non-trivial lower bound on the radius is known to the nodes, since we assume that they do not know the total number of nodes n.

B. The invariant

The main idea of the algorithm is that all nodes try to maintain the following invariant at all times:

Algorithm I
MainAlgorithm

```
1: me.radius ← CalculateRadius(me)
2: if me.role = open then
3:     if ∃ node v_j with [(v_j.role = open)
       ∧(v_j.radius ≤ me.radius)
       ∧(d(v_j, me) < 2 · me.radius)] then
4:         me.role ← closed
5:     end if
6: else
7:     if ∄ node v_j with [(v_j.role = open)
       ∧(v_j.radius ≤ me.radius)
       ∧(d(v_j, me) < 4 · me.radius)] then
8:         me.role ← open
9:     end if
10: end if
```

1) If $v_i \in C$, there is a facility $v_j \in F$ with $r_j \leq r_i$ and $d(v_i, v_j) \leq 4 \cdot r_i$
2) If $v_i \in F$, there is no other facility $v_j \in F$ with $r_j \leq r_i$ and $d(v_i, v_j) \leq 2 \cdot r_i$

As soon as a node discovers that its invariant is violated, it repairs it by changing its role. This can violate the invariant of other nodes. In section III-A we show that the effect of role changes is bounded.

C. The algorithm

As soon as a node v_i turns active, it calculates its radius (see below). Then it checks whether its invariant is fulfilled. If this is the case, it turns inactive again. Otherwise, its reaction depends on its role. If v_i is a facility, the invariant is violated because there is another facility with a smaller or equal radius within distance $2r_i$ of v_i. So closing v_i repairs its invariant. If v_i is a client, its invariant can only be violated because there is no facility with a smaller or equal radius within distance $4r_i$ of v_i. Therefore, in this case opening v_i repairs its invariant. Algorithm 1 formally describes the algorithm.

To calculate its radius, a node v_i computes k_1 as defined in Lemma 1 and then returns $k_0 = k_1 + 1$. Accordingly, its

Algorithm II
CalculateRadius

```
1: k_1^max ← ⌊log(f_i/c_i)⌋ + 1
2: V ← all nodes in distance at most 2^(k_1^max+1) to me
3: sort V_i by the distance to me in decreasing order
4: weight ← c_i
5: for all nodes v_j in V_i do
6:     weight ← weight + c_j
7: end for
8: k ← k_1^max
9: while weight ≥ f_i · 2^-k do
10:    for all nodes v_j in distance d, 2^(k-1) < d ≤ 2^k do
11:        weight ← weight −c_j
12:    end for
13:    k ← k − 1
14: end while
15: k_1 ← k + 1
16: return k_0 ← k_1 + 1
```

radius is 2^{k_0}. To compute k_1, v_i first computes the largest possible value for k_1, k_1^{max}, which is $\lfloor \log(\frac{f_i}{c_i}) \rfloor + 1$, and sets k to k_1^{max}. Then it sorts all nodes within distance $2^{k_1^{max}}$ by their distances to v_i, and computes the weight of the ball with radius $2^{k_1^{max}}$ as the sum of all c_j of nodes v_j within this distance. Now it reduces k one by one until the inequality weight$(B(v_i, 2^k)) \geq f_i \cdot 2^{-k}$ (see Lemma 1) is not valid any longer. To compute the current weight, the old value is decreased by the c_j of all nodes v_j which are in the ball with radius 2^{k+1}, but not in the ball with radius 2^k, using the sorting of the nodes. Since weight$(B(v_i, 2^k))$ is monotonically increasing and $f_i \cdot 2^{-k}$ monotonically decreasing with k, the resulting k is the largest value for which the inequality is not kept and therefore $k + 1$ the desired k_1. See Algorithm 2 for a formal description.

D. Events and initialization

To allow a better description of our algorithm and its analysis, we introduce the term *event*. An event occurs each time the radius of a node could change or an invariant could be violated because distances between nodes have changed. For a node v_i there are two reasons for an event to occur:

- The distance $d(v_i, v_j)$ between v_i and some node v_j changes in such a way that v_j enters or leaves the ball $B(v_i, \frac{1}{4}r_i)$ or the ball $B(v_i, \frac{1}{2}r_i)$. Here, the radius of v_i can change.
- The distance $d(v_i, v_j)$ between v_i and some node v_j changes in such a way that v_j enters or leaves the ball $B(v_i, 2r_i)$ or the ball $B(v_i, 4r_i)$. This can violate v_i's invariant, leading to a role change of v_i.

If one of these situation occurs, we say that v_j triggers an event at v_i. If v_i changes its role, invariants of neighboring nodes can be violated. A change of a radius can also affect other nodes: If the radius of a node v_i changes, it can become necessary for v_i to change its role as well. Moreover, if v_i increases its radius, there may be nodes which had a radius of r_i before and now have a smaller radius. If v_i is a facility, the invariants of those nodes can now be violated. The same can occur when v_i is a facility and decreases its radius: There can be nodes which had a smaller radius before and now have the same radius. Their invariants can now be violated, because they now take v_i into account. Note that if v_i is a client, changing its radius does not have any effects on the invariants of other nodes. Moreover, since the invariants only consider facilities and not clients, changing the radius of a facility can be viewed as two simultaneous role changes at the same position: The original node closes, and a new virtual node with the same position and the new radius opens. Thus, from now on we only consider role changes when we talk about events.

Often, events only affect one node directly and possibly other nodes indirectly. However, in some cases two nodes can be directly affected: if both nodes have the same radius, or one has the eightfold radius of the other. Even so, it can be shown that only one of these two nodes needs to change its radius or role. Thus, for the analysis we assume that an event only occurs at one node.

For our analysis, we can distinguish the *initialization*, which is the time until the invariants are kept for the first time at all nodes, and the time after that. After the initialization, changes of the role or radius can only occur because of events.

III. Analysis of the algorithm

Starting with general results about the algorithm we show that it performs well under dynamics, analyze the effects events can have and prove several lower bounds.

Theorem 1. *When the invariant holds for each node, the resulting set of facilities yields a 17-approximation.*

Proof: For each node v_i there is a facility v_j with radius $r_j \leq r_i$ in $B(v_i, 4r_i^*)$. Lemma 1 in combination with our definition of a radius yields $r_i^* \leq r_i \leq 4r_i^*$. Therefore $d(v_i, v_j) \leq 4 \cdot 4r_i^* = 16r_i^*$. The remaining of the analysis is analogous to the proof given in [12]. ∎

Theorem 2. *Each time a node v_i turns active, its local computations require $\mathcal{O}(n \log n)$ time. The data it needs to transmit is bounded by $\mathcal{O}(\log \log n)$ bits.*

Proof: The time for computing the radius is dominated by sorting the nodes within the maximum radius, which takes $\mathcal{O}(n \log n)$ time. Checking whether the invariant is fulfilled takes linear time only. The node v_i needs to transmit three pieces of information: its role, taking 1 bit, its c_i, which is constant, and its radius. Since there are only $\mathcal{O}(\log n)$ possible radii, the current radius can be transmitted using $\mathcal{O}(\log \log n)$ bits. We assume that each node knows the exact distances to all neighboring nodes within constant distance $(4 \cdot 4\frac{\max_{v_j} f_j}{\min_{v_j} c_j})$ and thus these distances do not need to be transmitted. ∎

Lemma 2. *A node v_i opening itself can only violate invariants of nodes with a strictly larger radius than r_i.*

Proof: Node v_i only opens itself if there is no other facility with a radius less than or equal to r_i within distance $4 \cdot r_i$ of v_i, since otherwise the invariant of v_i is not violated. When v_i opens itself, this does not affect nodes with a radius smaller than r_i due to the construction of the invariant. Moreover, if a node v_l also has radius r_i and is a facility, it must be in distance more than $4 \cdot r_i$ from v_i and therefore its invariant cannot be violated by v_i. If v_l is closed, a new facility does not violate v_l's invariant. ∎

From the next theorem follows that the initialization is efficient.

Theorem 3. *After $\mathcal{O}(\log n)$ rounds without an event, the invariant holds at all nodes.*

Proof: For the remainder of the proof we assume that no events occur. Thus, no node changes its radius. Consider a set of nodes S_r with $v_i \in S_r \Leftrightarrow r_i = r$ for a fixed radius r and a set $S_{<r}$ comprising all nodes with a radius smaller than r. Let t be the first round after which the invariant holds for all nodes with a radius smaller than r (i.e. those contained in $S_{<r}$). Since the invariants of the nodes belonging to $S_{<r}$ are not affected by nodes with a radius greater or equal r, starting from round $t+1$ none of these nodes will change its role. We say that these nodes are in a *stable* state, since their invariant can only be violated by the occurrence of an event. We now show that after round $t+2$ all nodes belonging to S_r have restored their invariant and are in a stable state. Consequently, after $2l$ rounds all invariants of nodes with one of the l smallest radii are in a stable state. Since $\mathcal{O}(\log n)$ is an upper bound on the number of different radii, the theorem follows.

Consider all nodes in S_r that are open directly before round $t+2$ begins. We claim that their invariants are not violated and that they will remain open (i.e. are in a stable state). To see this, we analyze the nodes' behavior during round $t+1$. There are two reasons for a node to be open at the end of round $t+1$: It was either closed before its activation in round $t+1$ and its invariant was violated, or it was open and did not need to change its role. In the first case, we know due to Lemma 2 that the closed node's change of role did not violate the invariants of any node in S_r and $S_{<r}$. Lemma 2 also guarantees that this node will never close again. In the second case, the already open node did not change its role and in accordance with Lemma 2 its invariant will not be violated in the following rounds.

Having established that all open nodes in S_r are in a stable state at the end of round $t+1$, we now need to show that all closed nodes in S_r are in a stable state at the end of round $t+2$. Note, that it is possible for a closed node in S_r to not be in a stable state at the end of round $t+1$, since all facilities within distance $4r$ around it might have closed after its activation in round $t+1$. When a closed node becomes active during round $t+2$ and its invariant is violated, it will open, not violate any invariant of nodes in S_r and $S_{<r}$ and stay open for the remaining rounds (due to to Lemma 2). On the other hand, when the invariant of an active closed node during round $t+1$ is not violated, it will stay closed for the remaining rounds, since it will never lose its facility (open nodes from S_r and $S_{<r}$ do not close in round $t+2$ or after it). This means that all nodes in S_r are in a stable state at the end of round $t+2$. ■

Corollary 1. *If no event occurs, the initialization takes* $\mathcal{O}(\log n)$ *rounds.*

We have proven that after $\mathcal{O}(\log n)$ rounds the nodes keep an $\mathcal{O}(1)$-approximation of the optimal solution until an event occurs. Before we analyze the effects of an event, we want to bound the number of events that can occur.

Theorem 4. *If the dynamics parameter in a time interval* $[t_0, t_1]$ *is upper bounded by* x *and all distances are in constant mode at time* t_0*, then the number of events in this time interval is* $\mathcal{O}(x \log n)$.

Proof: Consider a node v_i with radius r_i and another node v_j. If the distance between v_i and v_j remains the same (v_i and v_j are in constant mode), they cannot trigger events. If the distance decreases (they are in decreasing mode), v_j can trigger an event at v_i only when v_j enters a ball with center v_i and radius the $\frac{1}{4}$-, $\frac{1}{2}$-, two- or fourfold of one of the $\mathcal{O}(\log n)$ possible radii of v_i. Since v_j cannot leave a ball again as long as v_i and v_j stay in decreasing mode, v_j can trigger only $\mathcal{O}(\log n)$ events at v_i. Analogously, v_j can only trigger $\mathcal{O}(\log n)$ events at v_i when they are in increasing mode. Thus, for each change of mode, there are at most $\mathcal{O}(\log n)$ events resulting in $\mathcal{O}(x \log n)$ events in total. ■

Note that the $\mathcal{O}(\log n)$ factor in Theorem 4 is necessary, because the current radius of a node v_i can change. Especially, when v_j enters the ball with radius $\frac{1}{4}r_i$, r_i can decrease by one. Thus, when v_j enters the next smaller ball, this one again has radius $\frac{1}{4}r_i$ and r_i decreases. Thus, by decreasing the distance between v_i and v_j, v_i's radius can change from the largest possible to the smallest one, resulting in $\mathcal{O}(\log n)$ events.

A. Effects of events

Now we show that an event is handled efficiently. Especially, it affects only nodes in a constant distance from the event. We have already seen that it takes at most $\mathcal{O}(\log n)$ rounds until all invariants are kept again if no further event occurs in between. In a special geometric setting, even in the worst case only a polylogarithmic number of nodes can be affected by the event. Moreover, each affected node can change its role at most twice.

Theorem 5. *A node* v_j *can only be affected by an event if it is triggered at a node which is at most in distance* $12r_j$ *from* v_j.

Proof: Let v_k be a node at which an event is triggered. Thus v_k (and possibly also a virtual node at the same position) changes its role. We prove that only nodes v_j are affected by a role change of v_k for which hold $d(v_k, v_j) \leq 12 \cdot r_j$.

Let $e_i \cdot r_i$ be the maximal range around v_k in which nodes with radius at most $r_i = 2^i \cdot r_k$ are affected by the role change of v_k (note that r_i is not the radius of a specific node here). We show that $e_i \leq 12$, the theorem follows. We start with computing e_0. To do this, we first consider the case, where the role of v_k changes from open to closed. This does not affect nodes with radius less than r_k. Nodes with radius r_k, which are in distance at most $4 \cdot r_k$ from v_k, may now need to open themselves. Let v_l be such a node. Because of Lemma 2, no nodes with radius smaller than or equal to r_k close

themselves because of v_l. Thus, $e_0 = 4$. If the role of v_k changes from closed to open, the same argument applies: No node with the same radius r_k can close itself due to v_k. Thus, in this case $e_0 = 0$.

We now prove that $e_i = \frac{e_{i-1}}{2} + 6$ for $i > 0$. This applies independent from the type of role change of v_k. By the definition of e_{i-1}, nodes with radius $2^{i-1} \cdot r_k$, which changed their role due to the role change of v_k, can be at most in distance e_{i-1} times their radius from v_k. Let v_j be a node with radius $r_j = 2^i \cdot r_k = r_i$, which changes its role due to a role change of a node v_l with a smaller radius (if there is a node with radius r_j, which changes its role, there must also exist such nodes v_j and v_l). We first consider the case that v_j opens itself. Then, v_l must have closed itself and be within distance $4 \cdot r_j$ of v_j. Moreover, because of Lemma 2, no facility with radius less than or equal to r_j can close itself due to v_j. Because of the triangle inequality, v_j is in distance at most $e_{i-1} \cdot r_l + 4 \cdot r_j \leq (\frac{e_{i-1}}{2} + 4) r_j$ of v_k. The second case is that v_j closes itself. Now, v_l must have opened itself and be in distance of at most $2 \cdot r_j$ of v_j. Here, the role change of v_j may affect nodes with the same radius: a node v_m with radius r_j may have to open itself, because no facility is left within a range of 4 times its radius. Thus, v_m can be at most in distance $4r_m = 4r_j$ of v_j and therefore $6r_j$ of v_l (triangle inequality). Again, there cannot exist a node with radius less than or equal to $r_m = r_j$ which needs to close itself because of the role change of v_m. Thus, no node with radius less than or equal to $r_j = 2^i \cdot r_k$ which changes its role due to the role change of v_k can be in distance of more than $e_{i-1} r_l + 6 r_j \leq (\frac{e_{i-1}}{2} + 6) r_j$ of v_k and because $r_j = r_i$ and by the definition of e_i, $e_i \leq \frac{e_{i-1}}{2} + 6$.

This recurrence can be solved:

$$\begin{aligned} e_i \leq \frac{e_{i-1}}{2} + 6 &= \frac{e_0}{2^i} + \sum_{k=0}^{i-1} \frac{6}{2^k} \\ &= \frac{e_0}{2^i} + 6 \cdot \frac{1 - (\frac{1}{2})^i}{1 - \frac{1}{2}} \\ &= 12 - \frac{12 - e_0}{2^i} \leq 12 \end{aligned}$$

Thus, each node can only be affected by events at nodes within 12 times its radius. Note that this also holds if two role changes occur at the same time at the same position. ■

Since r_i is upper bounded by a constant ($4\frac{f_i}{c_i}$) and by Theorem 5 only nodes with distance $\leq 12 r_i$ to an event are affected by it, nodes can only be in constant distance from events by which they are affected.

Theorem 5 shows that executing our algorithm has benefits compared to executing some other $\Omega(\log(n))$-round algorithm ALG that is just invoked repeatedly and is not concerned with dynamics at all. While our algorithm guarantees that a new solution computed after an event occurred differs only slightly from the old one (only nodes within constant distance of the event change their role), the algorithm ALG might compute a new solution that is totally different from the old one.

Now we consider an important special case: nodes in Euclidean spaces. Additionally we have a slightly more restricted round model. We show that the number of affected nodes upon an event is bounded from above.

Theorem 6. *In a Euclidean space with constant dimension and for an asynchronous round model where each node turns active exactly once per round in an arbitrary order, an event affects at most $\mathcal{O}(\log^2 n)$ nodes, if no further event occurs before all invariants hold again.*

Proof: For simplicity we consider the case of the Euclidean plane only. For higher dimensions the arguments are analogous. We know from Theorem 3 after $\mathcal{O}(\log n)$ rounds all invariants hold again. It is left to show that in each round $\mathcal{O}(\log n)$ nodes change their role. Hence, we prove that for each of the $\mathcal{O}(\log n)$ possible radii at most a constant number of nodes change their role in each round. Consider one arbitrary round and a radius r_i. Let v_e be the node at which the event was triggered. If a node v_i has radius r_i, it has distance at most $12 \cdot r_i$ to v_e, due to Theorem 5. Therefore all nodes with radius r_i which potentially change their role are in an area around v_e with radius $12 \cdot r_i$ and therefore with an area $144\pi \cdot r_i^2$. On the other hand, at any time all facilities with radius r_i have a minimal distance of $2 \cdot r_i$ to each other leading to an area of $\pi \cdot r_i^2$ around each facility, which does not intersect with according areas of other facilities. Hence, at any given point in time, there are at most 144 facilities with a radius of r_i. Because each node is active only once, only the constant number of facilities at the beginning of the round can close. Out of the same reason, nodes which open themselves stay open until the end of the round, and so at most 144 nodes can open in one round. Thus, no more than 288 nodes with radius r_i can change their role in one round. For higher dimensions, we consider a d-dimensional ball of radius $12 \cdot r_i$ and bound the number of balls with radius r_i that fit into the volume. ■

Theorem 7. *For each event, each node changes its role at most twice.*

Proof: We show that when a node has closed itself, it only opens itself again because of a later event. From this follows the theorem.

Let v_i be a node that closed itself because of a chain of node role changes that was initially triggered by an event e. Because of Lemma 2, the node v_j that forced v_i to close itself must have a smaller radius than v_i, and it must lie within distance $2r_i$ of v_i. For v_i to open again, v_j must close. As long as no other event happens, this must be because another node v_l opens itself which is within distance $2r_j$ of v_j and for which $r_l < r_j$ (Lemma 2). This continues until we have no smaller radii left. Let v_k be the last node to open itself in this chain. We now show that it is still within

distance $4r_i$ of v_i.

We know that v_j is in distance at most $2r_i$ from v_i, $d(v_l, v_j) \leq 2r_j$ and so on. This yields the following sum as an upper bound on the distance that v_k can have from v_i:

$$\begin{aligned} 2r_i + 2r_j + 2r_l + \ldots &= 2r_i + 2 \cdot \frac{1}{2} r_i + 2 \cdot \frac{1}{4} r_i + \ldots \\ &\leq 2r_i \sum_{s=0}^{\infty} (\frac{1}{2})^s \\ &= 2r_i \frac{1}{1 - \frac{1}{2}} \\ &= 4r_i \end{aligned}$$

So as long as no new event occurs, there is always a facility with smaller radius within distance $4r_i$ of v_i and thus v_i does not close again. ∎

B. Lower bounds

Our algorithm shows that, in order to compute a constant factor approximation, it is sufficient for each node to have information about other nodes within a constant distance from itself. Thanks to [16] it is known that (unless $\mathcal{NP} \subset \mathcal{DTIME}(n^{\mathcal{O}(\log \log n)})$) it is not possible (even if each node is allowed to see the entire graph) to achieve a better than constant approximation for general metrics in polynomial time. (In the Euclidean case a (randomized) PTAS, which is based on the Arora scheme [17], is known [18]). Therefore, relaxing the locality constraints will not result in an improved approximation. Now, the question arises whether the approximation factor changes when the locality is restricted such that nodes are only able to see other nodes within a distance smaller than constant. The following theorem states that we lose in the approximation factor when locality is restricted in such a way.

Theorem 8. *A distributed algorithm ALG limited to information about nodes within distance $1/f(n)$ (in a general metric) can at best achieve an $\Omega(f(n))$-approximation for functions f with $\lim_{n \to \infty} f(n) = \infty$.*

Proof: Consider a star graph constructed in such a way that the distance from the center node v_0 to all other nodes is $\epsilon / f(n)$ for an $\epsilon > 1$. This star graph is now used to induce a metric M on the nodes. M is created by defining the distance between any two nodes as the shortest path in the star graph between them. We set $c_0 = f_0 = 0$ for the center node v_0 and $c_i = f_i = 1$ for the remaining nodes. Since an optimal algorithm OPT operating on M opens v_0 and keeps all other nodes closed, the resulting costs are 0 for the center node and $(n-1) \cdot \epsilon / f(n)$ for the remaining nodes.

Now, consider the metric M^∞ where all distances are set to infinity. Here, we also assign the $c_0 = f_0 = 0$ to v_0 and $c_i = f_i = 1$ to the remaining nodes. When operating on M^∞ any reasonable algorithm ALG is forced to open every single node, since closing at least one node raises the costs to infinity. Because all nodes are more than $1/f(n)$ away from each other and only have information about nodes within distance $1/f(n)$, ALG cannot distinguish the metric M from the metric M^∞ and therefore is forced to also open every single node when working on M. The ratio of the local algorithm ALG to the optimal algorithm OPT is

$$\frac{\text{ALG}}{\text{OPT}} = \frac{n-1}{0 + (n-1) \cdot \frac{\epsilon}{f(n)}} = \frac{f(n)}{\epsilon}$$

and thus $\text{ALG} \geq \frac{f(n)}{\epsilon} \cdot \text{OPT}$. ∎

We have shown that events only affect local neighborhoods and that the number of events can be upper bounded by $\mathcal{O}(x \cdot \log(n))$, where x is the dynamics parameter. In [4] it is shown that there exists a set of n nodes moving linearly on the real line that forces any c-approximate cover to change $\Omega(n^2/c)$ times. Setting all f_i and c_i to 1 yields that any c-approximation of the facility location problem in this setting has to change $\Omega(n^2/c)$ times as well. Therefore $\Omega(n^2/c)$ motion-induced role changes are inevitable. Since all nodes move linearly, for the dynamics parameter holds $x = O(n^2)$. Hence Theorem 4 is asymptotically tight up to the factor of $\log(n)$.

The next theorem states that in contrast to our approximation algorithm, which affects only a local neighborhood of constant size, an exact solution needs to change the role of nodes that are in linear distance from a single node which moves.

Theorem 9. *There exists a placement of nodes in the Euclidean plane such that the movement of a single node forces* exact facility location *to perform $\Omega(n)$ role changes. In particular, there are nodes which change their role in distance $\Omega(n)$ from the moving node.*

Proof: Consider a graph in the Euclidean plane with n nodes where $n = 3k + 4$ for some $k \in \mathbb{N}$. Let $c_i = f_i = 1$ for all i. The graph is a horizontal line where the distance between all nodes is $1 - \epsilon$ for some small $\epsilon > 0$, except for the nodes at the very right. Here, the last two nodes are arranged in such a way that they are in distance $1 - \epsilon$ to the third-last node and in distance $1 - \frac{3}{2}\epsilon$ to each other (and in distance larger than 1 to all other nodes, see figure 1). We will show that moving the leftmost node to the left for more than ϵ will change the exact solution for the facility location problem in $\Omega(n)$ nodes and hence covers an area, which has a linear stretch.

We now first show the optimal solution for the original graph. For the sake of analysis, we group the nodes in k components of 3 nodes each plus a component consisting of the 4 nodes at the right end. We show that choosing exactly the middle nodes in each component yields an optimal solution: This solution has facility costs of $k+1$ and client costs of $(2k+3) \cdot (1 - \epsilon)$, since each of the k facilities

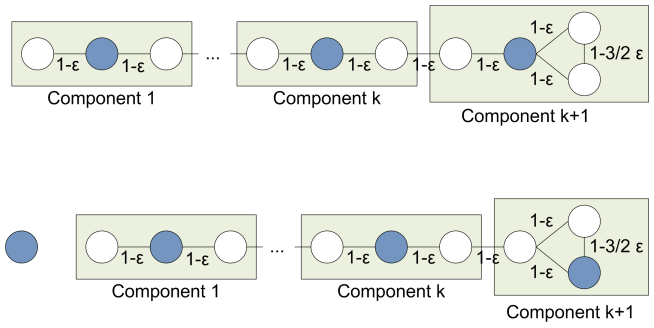

Figure 1. Graph for the proof of Theorem 9

produces a cost of 1 and each other node a cost of $1-\epsilon$. It is not possible to reduce the number of facilities, since in each component there needs to be at least one facility. Furthermore, the client costs can at most be reduced by $\frac{1}{2}\epsilon$ for choosing the connection to the right instead of another one plus $1-\epsilon$ times the number of additional facilities. Since opening additional facilities produces a cost of 1, this would increase the overall costs by ϵ for each additional facility. Furthermore, using the cheaper connection between the two rightmost nodes increase the number of facilities in this component by one. Since the number in the other components cannot be decreased, this would increase the overall costs by $\frac{\epsilon}{2}$. Thus the described choice of facilities is optimal.

Now we show that we have indeed $\Omega(n)$ role changes when moving the left node to the left. Eventually this node has to open. We arrange the remaining nodes in $k+1$ components again, but shift groups one unit to the right. The component on the very right now consists only of three nodes. With the same argument as above we need at least one facility in each component and it is cheapest to open the middle node in each except for the right one. Here it is now cheapest to open one of the nodes to the very right, since this does not increase the facility costs, but decreases the connection costs. The minimal costs in this scenario are therefore $(k+2)+(2k+1)\cdot(1-\epsilon)+1-\frac{3}{2}\epsilon$. For this operation, $2k$ nodes in the components have to change their role plus the node in the very left plus one of the two nodes in the right. ■

IV. Conclusion and Open Questions

We presented a simple $\mathcal{O}(1)$-approximation algorithm for the local facility location problem under worst case dynamics. We proved major key properties, such as upper bounds on the time until stabilization, the number of nodes affected by dynamics and - most important - that only a local neighborhood is affected by events. However, some questions remain open. An extension of our setting is the insertion and deletion of nodes. Furthermore, we assumed that all f_i and c_i are constant. It remains to show how our analysis changes when we allow those parameters to depend on n. The number of events in our setting only depends on the dynamics parameter. However, it would be better if many changes of direction on a short time interval would not trigger many events. The approximation factor depends on the number of possible radii. Is it possible to improve the approximation factor by considering more possible values for the radii? Does a trade-off exist between the number of events, the approximation ratio and the locality of the neighborhood? We need $\mathcal{O}(\log\log n)$ communication bits to guarantee a $\mathcal{O}(1)$-approximation. What approximation can be obtained with a constant number of bits? Moreover, we do not know yet whether Theorem 6 also holds in an unrestricted round model. Finally, we believe that our algorithm is resilient against all kinds of transient failures: several events at the same time, nodes that wake up at the same time, nodes missing some events or nodes that temporarily display wrong information.

References

[1] T. Moscibroda and R. Wattenhofer, "Facility location: distributed approximation," in *Proc. of the twenty-fourth annual ACM symposium on Principles of distributed computing (PODC)*, 2005, pp. 108–117.

[2] S. Pandit and S. V. Pemmaraju, "Return of the primal-dual: distributed metric facility location," in *PODC*, 2009, pp. 180–189.

[3] J. Gehweiler, C. Lammersen, and C. Sohler, "A distributed O(1)-approximation algorithm for the uniform facility location problem," in *Proc. of 18th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2006, pp. 237–243.

[4] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu, "Discrete mobile centers," in *Proc. of the seventeenth annual symposium on Computational geometry (SOCG)*, 2001, pp. 188–196.

[5] S. Bespamyatnikh, B. Bhattacharya, D. Kirkpatrick, and M. Segal, "Mobile facility location (extended abstract)," in *Proc. of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications (DIALM)*, 2000, pp. 46–53.

[6] B. Degener, J. Gehweiler, and C. Lammersen, "Kinetic facility location," *Algorithmica*, 2008.

[7] S. Har-Peled, "Clustering motion," *Discrete Comput. Geom.*, vol. 31, no. 4, pp. 545–565, 2004.

[8] C. Frank and K. Römer, "Distributed facility location algorithms for flexible configuration of wireless sensor networks," in *Distributed Computing in Sensor Systems*, 2007, pp. 124–141.

[9] S. Pandit and S. V. Pemmaraju, "Finding facilities fast." in *ICDCN*, ser. Lecture Notes in Computer Science, vol. 5408. Springer, 2009, pp. 11–24.

[10] D. Krivitski, A. Schuster, and R. Wolff, "A local facility location algorithm for sensor networks," in *Distributed Computing in Sensor Systems*, 2005, pp. 368–375.

[11] G. Wittenburg and J. Schiller, "A survey of current directions in service placement in mobile ad-hoc networks," in *Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2008, pp. 548–553.

[12] R. Mettu and C. Plaxton, "The online median problem," in *Proc. of the 41st Annual Symposium on Foundations of Computer Science*, 2000, pp. 339–348.

[13] M. Pal and E. Tardos, "Group strategy proof mechanisms via primal-dual algorithms," in *Proc. of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 2003, pp. 584–593.

[14] M. Badoiu, A. Czumaj, P. Indyk, and C. Sohler, "Facility location in sublinear time." in *Proc. of the 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, 2005, pp. 866–877.

[15] F. Mondada, A. Guignard, M. Bonani, D. Bär, M. Lauria, and D. Floreano, "Swarm-bot: From concept to implementation," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robot and Systems (IROS)*, 2003, pp. 1626–1631.

[16] S. Guha and S. Khuller, "Greedy strikes back: improved facility location algorithms," in *Proc. of the ninth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, 1998, pp. 649–657.

[17] S. Arora, "Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems," *J. ACM*, vol. 45, no. 5, pp. 753–782, 1998.

[18] S. Arora, P. Raghavan, and S. Rao, "Approximation schemes for euclidean k-medians and related problems," in *Proc. of the thirtieth annual ACM symposium on Theory of computing (STOC)*, 1998, pp. 106–113.