COMPUTER SCIENCE REVIEW 5 (2011) 57–68

available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cosrev

Computer Science Review

Survey

A survey on relay placement with runtime and approximation guarantees

Bastian Degener[a], Sándor P. Fekete[b], Barbara Kempkes[a,*], Friedhelm Meyer auf der Heide[a]

[a] Heinz Nixdorf Institute, CS Department, University of Paderborn, 33098 Paderborn, Germany
[b] Braunschweig University of Technology, IBR, Algorithms Group, 38106 Braunschweig, Germany

ARTICLE INFO

Article history:
Received 29 June 2010
Received in revised form
20 August 2010
Accepted 15 September 2010

Keywords:
Relay placement
Mobile robots
Local algorithms
Robot formations

ABSTRACT

We discuss aspects and variants of the fundamental problem of *relay placement*: given a set of immobile terminals in the Euclidean plane, place a number of relays with limited viewing range such that the result is a low-cost communication infrastructure between the terminals. We first consider the problem from a global point of view. The problem here is similar to forming discrete Steiner tree structures. Then we investigate local variants of the problem, assuming mobile relays that must decide where to move based only on information from their local environment. We give a local algorithm for the general problem, but we show that no local algorithm can achieve good approximation factors for the number of relays. The following two restricted variants each address different aspects of locality. First we provide each relay with knowledge of two fixed neighbors, such that the relays form a chain between two terminals. The goal here is to let the relays move to the line segment between the terminals as fast as possible. Then we focus on the aspect of neighbors that are not fixed, but which may change over time. In return, we relax the objective function from geometric structures to just forming a single point. The goal in all our local formation problems is to use relays that are as limited as possible with respect to memory, sensing capabilities and so on.

We focus on algorithms for which we can prove performance guarantees such as upper bounds on the required runtime, maximum traveled distances of the relays and approximation factors for the solution.

1. Introduction

Because of the increased availability of cheap mobile computing entities, algorithmic research focusing on sensor networks and groups of autonomous robots has grown considerably. In many of these systems, the computing entities need to interact, but are limited to do so only with nearby entities. One possible solution is to provide *relays* that build a communication infrastructure between the computing entities. These relays are also limited to local interactions. Therefore, a central goal is to build a low-cost infrastructure, using as few relays as possible, to connect a given set of computing

* *Corresponding author.*
E-mail addresses: degener@upb.de (B. Degener), s.fekete@tu-bs.de (S.P. Fekete), barbaras@upb.de (B. Kempkes), fmadh@upb.de (F.M. auf der Heide).

1574-0137/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.cosrev.2010.09.005

entities (*terminals*). Since the relays should be as cheap as possible, we want to restrict them as much as possible in terms of memory, communication devices, sensing capabilities, etc.

Due to their distributed nature, many of these systems do not have a central controller that knows the whole state of the system; therefore, relays cannot be placed centrally. Thus, our goal is to design local strategies that nevertheless lead to provably good global solutions. To achieve this, the relays are mobile and must determine where to move based solely on the information they can gather from their local surrounding. In particular, they may not communicate with each other (they can just relay data between terminals), but they have a viewing range in which they can determine positions of relays and terminals (together referred to as *robots*) relative to their own position. In addition, relays and terminals have a communication range (in most of our concrete scenarios, the communication and viewing range will be the same).

Formally, let $C = (V_C, E_C)$ be the communication graph, where V_C contains a vertex for each relay and terminal; E_C contains an edge for every pair of relays/terminals that are in mutual communication range. The *Sparse Communication Network Problem*, described informally above, can now be formulated as follows. Given m terminals and n mobile relays in the plane, such that the communication graph is connected, move the relays in the plane such that the communication graph stays connected and the number of relays is minimized. To achieve this, relays may remove themselves from the system (this can be seen as the return to a relay depot).

Within this survey, we will furthermore consider two settings that simplify the general problem in two ways. For the *Short Chain Problem*, we restrict the number of terminals to two and consider a chain of relays between these terminals. The chain may be arbitrarily winding in the beginning, and the goal is to move the relays to the line segment between the two terminals, still keeping the communication graph connected. Here, we provide the relays with information about their predecessor and successor. This way we omit the problem that the number of relays in the viewing range in the general setting can be large and that these neighbors can change. In the second setting, we capture the problem of large and changing neighborhoods, but instead we drop the problem of forming short communication structures. In the resulting setting, known as *gathering*, we do not have terminals but just n relays. The goal is to gather the relays in one single point. In both of these settings, the challenge is to coordinate the relays when restricted to a purely local view.

The first obvious question is whether these three problems can be solved globally and what the solutions look like. For the Gathering Problem, a global solution is simple: compute one point and position all relays there. Similarly, the Relay Chain Problem can be solved globally by positioning all relays on the line segment between the terminals, making sure that the communication graph is connected. For the Sparse Communication Network Problem, a global solution is not obvious. In Section 3.1 we will see that the problem is NP-hard and therefore analyze it with respect to its approximability.

Then we will focus on the local aspects of the problem. We present a local algorithm for the Sparse Communication Network Problem that reduces the number of relays. It can be seen that local strategies cannot guarantee good approximation factors for the number of relays for this problem. We therefore concentrate on the two refined settings and show approximation algorithms with runtime guarantees.

We first give algorithms for the Relay Chain Problem and analyze them with respect to the number of time steps until an approximation of the line is achieved (Section 4). It turns out that the number of time steps depends on the length of the steps (which is the maximum distance covered by a relay in one time step), and thus we analyze this dependency in a separate section (Section 5). Then we proceed to the Gathering Problem (Section 6). Again, we give an algorithm and analyze it with respect to the number of time steps until gathering is achieved.

For each refined setting, we first provide an intuitive algorithm and analysis. Then we show how a more complex algorithm with a more involved analysis leads to better runtime bounds. The analysis is backed by lower-bound considerations. This is a rather unusual approach, since most related work considers only feasibility within finite time, while we explicitly focus on runtime bounds.

When considering mobile robots, the typical research goal is to find the minimal robot abilities such that the robots can still perform the given task. This is also the approach we want to take. We will therefore concentrate on very weak relays.

Robot models. In the literature, different robot abilities/ constraints are distinguished. One fundamental ability of robots is to observe the positions of the other robots in their own local coordinate system. Possible variants arise from robots using a common coordinate system, robots using the same directions but different units of measurement, and robots having different local coordinate systems; in addition, there are intermediate models. In this paper we will focus on robots with different local coordinate systems.

The most restrictive robot constraint is that robots are *oblivious*, i.e., at a given time t they do not know anything about their state any time $t' < t$. In addition, they are often assumed to be *anonymous*, that is robots do not have IDs and cannot be distinguished from each other. Mostly, robots are assumed to be *point robots*, i.e., they do not have an extent and are positioned on a single point in the plane. Moreover, they can share this position with other robots.

Robots are often prohibited to communicate explicitly. Implicit communication is performed by moving to a new position (*interaction via sensing*).

With these restricted abilities, robots often act in *Look-Compute-Move (LCM)* steps: they first observe the positions of the other robots, then they compute a new position (referred to as *target position*) and finally they move to the computed position.

The focus in this paper is on locality. To model this, we assume that the robots have a constant *viewing range*, and they can only observe the positions of robots within that range. In combination with oblivious robots, this means that in each time step, the robots only know the positions of those robots within their viewing range and thus their new position only depends on this data. Note that in this setting, communication is not a powerful tool, because information can only be passed to those robots within the sender's transmission range (which we mostly assume to be equal to

the viewing range). In the next time step, the oblivious robots do not remember the information they got in the previous step and thus cannot pass it further.

Furthermore, if we want to build communication infrastructures with as few relays as possible, relays need to be able to *remove* themselves from the system if they are not needed. This can be seen as returning to a relay deposit. If relays do not have this ability, the most feasible objective is to minimize the length of the communication infrastructure in terms of the sum of the distances between the relays on shortest paths between terminals. An equivalent notion to deletion is to *fuse* robots. Here we assume that two or more robots move to the same position and never part again. Note that deleting relays is only possible if the network connectivity is still guaranteed. For instance in the next paragraph we will introduce a synchronous time model, where this condition is typically not fulfilled. Hence, in the remainder of this paper we consider deleting relays only if applicable.

Time models. Similar to the robot models, different time models can be distinguished. One of these is the *synchronous* time model. Here, all robots perform the LCM-steps at the same time. This means that all robots use the current positions to compute their new position, and then all robots move at the same time. We refer to one LCM-cycle as a *time step*. When referring to positions of robots in a time step, we mean the positions during the Look and the Compute steps.

Different *asynchronous* models exist in addition to the synchronous model. The simplest of those assumes that only one robot is active at a time, where the order of activation may be determined by an adversary. This model avoids the problem of symmetry breaking: if several robots act at the same time, each single operation may be good, but the combination results in a bad or infeasible configuration. In our scenario, one such problem is that the network can break apart if several robots move at the same time. When using a synchronous model, it must explicitly take care of symmetry breaking, while an asynchronous model can focus on the actual algorithm. On the other hand, a risk of the asynchronous model is to use hidden global knowledge. Thus, we will only use this model in scenarios in which only *local* symmetry breaking is needed: a robot may not move at the same time as any of its neighbors in its local viewing range, while the movement of robots far away is no problem. If a synchronous model is used, this can also be solved locally in one communication round after the Look step of LCM by using randomization or IDs. To measure time in this asynchronous model, a typical approach is to count *rounds*. A round ends as soon as all robots have been active at least once. A *time step* in this model is the time in which one single robot is active.

In this paper, a focus is on analyzing algorithms with respect to the runtime. When not considering runtime, there exist some more time models as generalizations of the synchronous model. In the *semi-synchronous* model, in each time step only a subgroup of robots is active and performs the LCM-steps. For the sake of worst-case analysis we assume that the subgroups are chosen by an adversary. In a general *asynchronous* model, in each time step a subgroup of robots is active performing either a Look, a Compute or a Move step. That is, when performing a Move step, the data used for computing the target position may be out of date. Some authors even assume that the target position may not be reached during the first Move step. These models are so far only used for analyzing whether the robots can solve a task in finite time. This is why in this paper we use the synchronous and the simple asynchronous model or variants of them.

Related work. The relay placement problem in its general form has been considered from a centralized global point of view. Because it is $\mathcal{NP}$-hard, it is interesting to consider polynomial-time approximation algorithms. This was done by Lloyd and Xue [1], who gave an algorithm with an approximation ratio of 7 for one-tier relay placement, and a $(5+\varepsilon)$-approximation algorithm for an arbitrary viewing range of at least 1. Srinivas et al. [2] gave a $(4+\varepsilon)$-approximation for a viewing range of at least 2. See references in [1,2] for earlier works.

The currently best results were achieved by Efrat et al. [3], who gave a simple 6.73-approximation algorithm and a polynomial-time 3.11-approximation algorithm for the one-tier version. Moreover, they showed that there is no PTAS for one-tier relay placement (assuming that the viewing range is part of the input, and P $\neq$ NP), and they gave a PTAS for two-tier relay placement.

The local formulation of the relay placement problem is closely related to robot formation problems, which mainly have been considered in parallel models, but under a global view. The general goal is to let robots, which are distributed in the Euclidean plane, build a formation, for example a circle [4,5].

The Gathering Problem, considered in Section 6, is one of the most important formation problems. The goal is to let the robots gather in one point in the plane, which is not predetermined. Due to its simplicity the problem has been intensively studied. The main focus was – similar to this paper – to pinpoint the crucial robot capabilities in a given time model in order to solve the problem at hand, either by an algorithm or by impossibility results [6–10]. However, there are few works on robots that are limited to a local view and also there is little known about runtime bounds. Usually the goal is only to gather in finite time. Notable exceptions are [11–13]. The authors of [11] show that convergence to a point (an easier variant of the Gathering Problem) is possible with time $\mathcal{O}(n^2)$ until the diameter of the convex hull of robots is halved, but the robots need a global view. [13] solves the robot convergence and [12] the robot gathering problem locally, both without giving runtime bounds. Both algorithms basically let a robot move to the center of the smallest enclosing circle around its neighbors. We will give details on [12] in Section 6. Neither the Short Chain Problem nor the general Sparse Communication Network Problem have been considered from a local point of view in works not covered by this paper.

Overview. We tackle the problem of relay placement in several steps. First we consider the general variant of the problem, the Sparse Communication Network Problem, in Section 3. We provide global approximation algorithms, before we present first results for a local strategy. The analysis for the local case captures important aspects concerning resulting configurations, but does not provide a runtime guarantee. On the one hand, problems stem from the fact that

structures are not easy to form locally, on the other hand from changing neighborhoods. In Section 4 we focus on the former, and analyze strategies for the Short Chain Problem using relays which are provided with information about their neighborhood relationship. This approach is further refined in Section 5, where important quality measures for local approximation algorithms are analyzed in detail, such as the length of the traveled path. In Section 6 we follow the other approach and focus on changing neighborhoods, but omit the problem of building structures by investigating the problem of gathering robots in one point. Finally we conclude in Section 7. In each section we first formally define the problem and the model. Then we describe algorithms and state results of the analysis with an intuition why the results hold.

2. Preliminaries and notation

In the following, we will denote by n the number of relays and by m the number of terminals. Relays and terminals together will be referred to as *robots*. The relays are numbered from 1 to n (while these IDs are not known to the robots), the position of relay r_i is denoted as p_i. The positions of relays and terminals define a *configuration* in a time step t. Accordingly, the *start configuration* or *initial configuration* describes the positions of relays and terminals at the beginning, the *final configuration* is the configuration in which the robots stop moving or to which they converge. Since our robots have a limited viewing range, in each time step they can only see a subset of the robots. $N_t(i)$, the neighborhood of relay r_i at time t, comprises all robots which r_i can see at time t excluding r_i itself.

In some sections, we will use the convex hull of the positions of robots. For simplicity, we will just call it the convex hull of the robots. $|x, y|$ will denote the Euclidean distance between two points x and y, for the distance between the positions of two robots r_i and r_j we will also use $|r_i, r_j|$. During one Move step, robots move up to a certain distance, which will sometimes be bounded explicitly. If so, we refer to the maximum distance a relay may cover in one Move step as the *step length*.

Preliminaries. Local algorithms must make sure that the whole network stays connected, because if connection is lost, the robots may not be able to connect the network again. More precisely, consider the graph with one vertex for each robot and an edge between every two robots which are in viewing range of each other, which is also known as unit disk graph. This graph must always stay connected. All our local algorithms guarantee this.

Theorem 2.1. *All algorithms presented in this paper ensure that a network that is connected in the beginning remains connected during the runtime of the algorithm.*

For the global algorithm this is not an issue. Considering the local strategies this is usually easy to show due to the definition of the respective strategy.

3. The sparse communication network problem

In this section, we consider the problem of connecting terminals by small relay networks. Given a set of m terminals in the plane, the goal is to place relays in the plane such that the communication graph of terminals and relays is connected, while the number of relays is minimized. We first consider this problem in a global and centralized scenario: a central unit which knows the whole situation may place relays arbitrarily to solve the problem. Even in this setting, the problem is NP-hard. We give an approximation algorithm to solve the problem. Then we take a look at the local variant of the problem. Given a set of m terminals and a set of n relays in the plane, each with a local viewing and transmission range of 1, how should the relays move to allow as many relays as possible to remove themselves? We describe a strategy for this problem here and analyze the resulting configurations.

3.1. Global view

From a global point of view, relay placement consists of finding a small set of locations, such that a given set of points can be connected. It is not hard to see that this contains the NP-hard problem of computing a minimum Steiner tree as a special case, if the distances between terminals are relatively large when compared to the terminals' transmission range. This makes it interesting to consider polynomial-time approximation algorithms.

The motivation of forming communication structures demands for a model how the terminals can communicate.[1] For this, two models have been considered in the literature [1,14–16,2,17,18]. In both models, a terminal and a relay can communicate if the distance between them is at most 1, and two relays can communicate if the distance between them is at most r, where $r \geq 1$ is a given number. The models differ in whether direct communication between terminals is allowed. In the *one-tier* model two terminals can communicate if the distance between them is at most 1. In the *two-tier* model the terminals do not communicate at all, no matter how close they are. In other words, in the two-tier model the terminals may only link to relays, but not to other terminals.

Formally, the input to the relay placement problem is a set of m terminals, identified with their locations in the plane, and a number $r \geq 1$, the communication range of a relay which we will also call *viewing range* (w.l.o.g. the communication range of a terminal is 1). The objective in the *one-tier* relay placement is to place a minimum number of relays so that between every pair of terminals there exists a path, *through terminals and/or relays*, such that the consecutive vertices of the path are within distance r if both vertices are relays, and within distance 1 otherwise. The objective in the *two-tier* relay placement is to place a minimum number of relays so that between every pair of terminals there exists a path *through relays* such that the consecutive vertices of the path are within distance r if both vertices are relays, and within distance 1 if one of the vertices is a terminal and the other is a relay (going directly from a terminal to a terminal is forbidden).

We will now present the results from [3]. The main approach for the approximation algorithms consists in separating local connectivity of clusters of terminals from the global connectivity of the resulting connected components. We start by describing the former, and then sketch the latter.

[1] This is just a motivation for the problem. For our algorithms which *form* the structures no communication is needed.

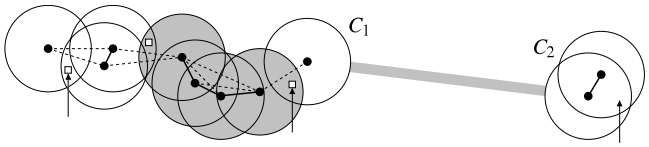

Fig. 1 – Dots are terminals in V, solid lines are edges in E and F, and dashed lines are edges in F only. There are 5 blobs in $\mathcal{B}$ (one of them highlighted) and 2 clouds $C_1, C_2 \in \mathcal{C}$. Arrows indicate a set of relays that hit all blobs, while small rectangles connect all blobs. The wide grey line is the only edge in MStFN($\mathcal{C}$), which happens to be equal to MSFN($\mathcal{C}$) here.

3.1.1. *Local connectivity*

Let V be a given set of terminals (points in the plane). We form a unit disk graph $\mathcal{G} = (V, E)$ and a disk graph $\mathcal{F} = (V, F)$ where $E = \{\{u, v\} : |uv| \leq 1\}, F = \{\{u, v\} : |uv| \leq 2\}$; see Fig. 1.

A *blob* is defined to be the union of the unit disks centered at the terminals that belong to the same connected component of $\mathcal{G}$. We use B to refer to a blob, and $\mathcal{B}$ for the set of all blobs.

Analogously, a *cloud* $C \in \mathcal{C}$ is the union of the unit disks centered at the terminals that belong to a connected component of the graph $\mathcal{F}$. The terminals in a blob can communicate with each other without relays, while the ones in a cloud might not, even though their disks may overlap. Each cloud $C \in \mathcal{C}$ consists of one or more blobs $B \in \mathcal{B}$; we use $\mathcal{B}_C$ to denote the blobs that form the cloud C.

Placing at least one relay in each blob of a cloud is a necessary condition for connecting them; a key observation is the following straightforward lemma, which allows actually achieving connectivity at not too high a cost:

Lemma 3.1. *Given a set S of relays such that each blob in a cloud $\mathcal{B}_C$ contains a relay $s \in S$, we can obtain in polynomial time a set of not more than $2|S| - 1$ relays that connect all terminals in $\mathcal{B}_C$.*

Making use of this lemma requires using an appropriate set-cover method for computing an approximately optimal set S of relays. Because no relay can be part of more than five different blobs, a simple greedy set cover approach yields an approximation factor of $1+1/2+1/3+1/4+1/5 = 137/60$. As a result of Lemma 3.1, we get an approximation factor of $137/30$ for making the set of terminals in a cloud connected.

3.1.2. *Global connectivity*

The idea for globally connecting all clouds is to use appropriate graph structures. For a collection $\mathcal{P}$ of planar subsets called the *neighborhoods*, the *Minimum Steiner Forest with Neighborhoods* on $\mathcal{P}$, denoted MStFN($\mathcal{P}$), is a *minimum-length* plane graph G such that $\mathcal{G}_{\mathcal{P}} = (\mathcal{P}, E(G))$ is connected. The MStFN is a generalization of the Steiner tree of a set of points. Note that MStFN is slightly different from a Steiner tree with neighborhoods (see, e.g., [19]) in that we are only counting the part of the graph *outside* each neighborhood $\mathcal{P}$ towards its length (since it is not necessary to connect neighborhoods beyond their boundaries).

Consider a complete weighted graph whose vertices are the neighborhoods in $\mathcal{P}$ and whose edge weights are the distances between them. A minimum spanning tree in the graph is called the *Minimum Spanning Forest with Neighborhoods* on $\mathcal{P}$, denoted MSFN($\mathcal{P}$). A natural embedding of the edges of the forest is by the straight-line segments that connect the corresponding neighborhoods; we will identify MSFN($\mathcal{P}$) with the embedding. (As with MStFN, we count the length of MSFN only *outside* $\mathcal{P}$.)

We denote by $|\text{MStFN}(\mathcal{P})|$ and $|\text{MSFN}(\mathcal{P})|$ the total length of the edges of the forests. It is known that $|\text{MSFN}(P)| \leq (2/\sqrt{3})|\text{MStFN}(P)|$ for a *point* set P, where $2/\sqrt{3}$ is the *Steiner ratio* [20]. The following lemma generalizes this to neighborhoods.

Lemma 3.2. *For any $\mathcal{P}$, $|\text{MSFN}(\mathcal{P})| \leq (2/\sqrt{3})|\text{MStFN}(\mathcal{P})|$.*

As a result, it is possible to give a simple polynomial-time algorithm with approximation factor $6.73 > 137/30+1+2/\sqrt{3}$.

Theorem 3.3 ([3]). *There is a simple 6.73-approximation algorithm for one-tier relay placement.*

3.1.3. *A refined method*

Replacing the greedy approach for local connectivity by a more refined approach, in combination with more careful and separate bookkeeping for local and global connectivity, leads to the following result.

Theorem 3.4 ([3]). *There is a 3.11-approximation algorithm for one-tier relay placement.*

The basic steps of the algorithm are as follows:

1. Compute optimal stabbings for clouds which can be stabbed with few relays.
2. Connect the blobs in each of these clouds, using Lemma 3.1.
3. Greedily connect all blobs in each of the remaining clouds ("stitching").
4. Greedily connect clouds into clusters, using 2 additional relays per cloud.
5. Connect the clusters by a spanning forest.

3.1.4. *Further results*

As it turns out, one-tier and two-tier characteristics are fundamentally different:

Theorem 3.5 ([3]). *It is NP-hard to approximate one-tier relay placement within a factor $1 + 1/687$, provided that r is part of the input.*

Theorem 3.6 ([3]). *There is a polynomial-time approximation scheme (PTAS) for two-tier relay placement.*

The proof for Theorem 3.5 is based on a reduction from the inapproximable problem VERTEX COVER. The PTAS makes use of the approach of *guillotine subdivisions*, which were introduced by Mitchell [21].

3.2. *Local view*

To analyze the effects of locality, we want to use a local model which is as restricted as possible. For simplicity, we set $r = 1$

and thus the communication range of all robots is 1. We also set the viewing range to 1. In addition to this, our relays are prohibited to exchange information about the network. This is modeled by two robot constraints (as described in *robot model*). First, our relays are oblivious, that is, at a given time t they do not know anything about any time $t' < t$. Secondly, our relays are not allowed to use communication to exchange information. With these constraints, the relays can only pass on information by occupying a certain position. Further robot properties are that they are anonymous, they are point robots and they can remove themselves. We use the general asynchronous model as described in Section 1 as a time model. The results in this section were published in [22,23].

We will now first describe the algorithm. Then we will give an overview about the resulting configurations with several terminals and in the special setting of two terminals. This setting is closely related to the Short Chain Problem which is described in Section 4.

The Go-To-The-Center*-strategy.* If relay r_i is active in time step t, it first observes the exact positions of all its neighbors ($N_t(i)$). It then checks if the subgraph of the unit disk graph induced by $N_t(i)$ is connected. If this is true, r_i removes itself. Otherwise, it computes its new position as the center of the smallest enclosing circle around all positions of its neighbors within its viewing range. This center is equivalent to the point that minimizes the maximum distance to the nodes in $N_t(i)$. Note that this algorithm keeps the network connected because the moving relay minimizes the maximum distance to its neighbors and so it does not increase the distance to the neighbor which is furthest apart. Thus, neighboring relays stay neighbors until one of them is removed.

Analysis with several terminals. We will now give an insight into the analysis of the Go-To-The-Center-strategy in settings with m terminals, before we investigate the special case of $m = 2$. Especially, we want to know which kinds of configurations are achieved when applying the Go-To-The-Center-strategy. Runtime bounds are not known for this problem.

Theorem 3.7 ([22]). *When applying the* Go-To-The-Center*-strategy, all relays will eventually be within ϵ-distance to the convex hull of the terminals, for every $\epsilon > 0$.*

Although Theorem 3.7 states that the area which may be covered by relays is reduced to the convex hull of terminals, the next theorem shows that there may nevertheless be a large number of relays, which is quadratic in the largest distance between the terminals.

Theorem 3.8 ([23]). *Let d be the maximal distance between two terminals. There are configurations with $\Theta(d^2)$ respectively $\Theta(m \cdot d)$ relays which cannot be changed by the* Go-To-The-Center*-strategy. Optimal configurations use $\Theta(d)$ relays for these configurations.*

Note that the lower bound of $\Omega(m \cdot d)$ only dominates iff m is at least of the order of d. This on the other hand implies that there is a large number of terminals in a very small area, which appears to be a rather artificial situation.

For the configuration with $\Theta(d^2)$ relays, position four terminals in a square with diagonal d. Then place the relays on a grid between the terminals with grid width w, $\frac{1}{\sqrt{2}} < w \leq 1$. In this configuration, each relay can see its four neighbors on the grid, but no relay can detect that it is positioned on a circle in the unit disk graph. Thus, each relay must stay where it is in order to guarantee global connectedness of the network. No local strategy can therefore improve this configuration. If we want to reduce the number of relays, we must relax the locality constraints and therefore change the robot model. Then the problem becomes easy: allowing communication with non-oblivious robots enables a distributed DFS or BFS to obtain a tree of relays connecting the terminals.

Analysis with two terminals. If we restrict the number of terminals to two, due to Theorem 3.7 the relays converge to the line between the terminals. In this case, circles in the unit disk graph of relays and terminals can only occur in complete subgraphs. Using this, the number of relays which Go-To-The-Center finally achieves can be upper-bounded.

Theorem 3.9 ([23]). *If d is the distance between the two terminals,* Go-To-The-Center *reduces the number of relays to $2d - 2$.*

Thus in this setting, Go-To-The-Center achieves a 2-approximation of the number of relays used in an optimal setting. Runtime bounds are not known, and thus we will consider a similar but simplified variant in the next section.

4. The Short Chain Problem

In this section we focus on one aspect of the Sparse Communication Network Problem which is difficult to achieve under locality: building communication structures. More precisely, we want to build a straight line between two terminals. On the other side, we want to avoid the problem rising from changing neighborhoods. To achieve this, we provide the relays with local neighborhood relationships. More precisely: we are given two terminals with fixed positions in the plane and an arbitrary winding chain of relay robots in between. Each relay r_i has two neighbors r_{i-1} and r_{i+1} in the chain, which are in viewing range (distance 1) of r_i, and which can be distinguished from the other robots by r_i. Again, we assume a communication range of 1. For the Short Chain Problem, the goal is to minimize the sum of the distances between adjacent robots (relays and terminals) in the chain, using only local information. In each time step, each relay robot knows only the relative position of its chain neighbors. From a global point of view this is a trivial problem, because the relays only have to be placed on the straight line between the two segments. However, the problem turns out to be challenging when considered locally.

First we review a simpler version of the Go-To-The-Center-strategy from the last section, which we call Go-To-The-Middle [24–26]. Here, a relay robot moves to the center between its two neighbors in synchronous rounds. We show that this strategy converges towards the optimal solution, but requires quadratic time. Based on this result we propose a linear time algorithm in a slightly weaker model, called the Hopper-strategy [27,26].

4.1. *The* Go-To-The-Middle *strategy*

Model. We consider the synchronous time model and a robot model which is similar to the model in the last section. In this section we do not allow robots to remove themselves, since this always bears the risk of losing connectedness in a synchronous time model. We provide the relays with their neighborhood relation as described above, but they are not able to distinguish which robot is the predecessor and which the successor.

The main constraint for a strategy is the fact that the chain is not allowed to break. Therefore the relay robots have to move such that at the end of the time step the chain is still connected.

The strategy. The following Go-To-The-Middle strategy is executed repeatedly by every relay robot in every time step. Relay robot r_i observes the positions of its neighbors in the chain and moves itself into the middle of the line segments of its neighbors. According to the LCM-model, all relays first check the positions of their neighbors. Then they compute their target position as described above and move there.

For this strategy, a runtime bound can be proven:

Theorem 4.1 ([24]). *Consider an initial chain with $n-2$ relay robots. When applying* Go-To-The-Middle*, after at most $9n^2 \log \frac{1}{\epsilon} n$ time steps holds for every $r_i d(p_i) \leq \epsilon$ for any $\epsilon > 0$, where $d(p_i)$ is the distance of relay r_i to its final position.*

The main idea of the proof is to consider the vector of distances to the positions on the final line and formulate the Go-To-The-Middle strategy as a matrix operation on this vector. The resulting matrix is symmetric, substochastic and irreducible. The eigenvalues of the matrix can be computed and using results from Markov Chain theory follows a convergence rate as stated in Theorem 4.1.

On the other hand this bound is almost tight:

Theorem 4.2 ([26]). *There exists an initial chain of relays such that the* Go-To-The-Middle *strategy needs $\Omega(n^2)$ steps in order to reach a maximal distance of 1 from the final position for all relays r_i, $i = 1, \ldots, n$.*

The idea is to consider a scenario with a triangular shape. The uppermost robot has to move the furthest. The movements can be explicitly analyzed and lead to the bound mentioned above.

Thus, the most intuitive strategy does not perform very well: note that a trivial optimal strategy with global knowledge would require only a linear number of steps in the worst case, since no relay can be in more than linear distance from the terminals.

4.2. Hopper

The above results give rise to the Hopper-strategy considered in [27,26], which has a linear runtime. Here, the model is slightly weaker, because we have a round model that gives the opportunity to know whether a neighbor was active in the last step and it is allowed to remove relays from the chain. Furthermore, the defined goal is easier to be achieved: we only require the length of the chain to be a constant-factor approximation of the optimal chain. This slight strengthening of the model allows us to formulate an algorithm that is closely related to Go-To-The-Middle, but more involved and with only a linear runtime.

Model. We extend the model such that a relay can remove itself. Furthermore, it can distinguish its left and right neighbor and knows which of them have moved in the last time step and which operations the neighbors have just performed (which leads to non-oblivious robots in the strict sense). Note that this fact leads to the possibility of executing sequential runs: a relay at the left end of the chain can start to move. Based on this movement the next relay in the chain can move in the next time step and so on. This leads to the notion of runs: the performance of an operation of the leftmost relay passes through the entire chain and each movement can be interpreted as the result of this movement, which helps for the analysis. Note that runs can be pipelined by starting a run in every third time step.

The strategy. For describing the strategy, let p_i denote the position of relay r_i at the beginning of a run and p'_i its position at the end of the run. Its left neighbor (and therefore predecessor) is r_{i-1}.r_i thus moves one time step after r_{i-1} has moved. Accordingly r_i's successor is r_{i+1}.

The behavior of relay r_i in the run depends on whether the distance between p'_{i-1} and p_{i+1} is larger than 1 or not. If the distance is smaller than or equal to 1 then obviously r_i is no longer necessary in the chain. In this case the relay removes itself, hereby executing the *remove-operation*. Executing the remove-operation finishes the run of the Hopper strategy: the following relays do not move in this run. In case the distance between p'_{i-1} and p_{i+1} is larger than 1, the action of r_i depends on the angle between the line segments $\langle p'_{i-1}, p_i \rangle$ and $\langle p_i, p_{i+1} \rangle$. The hop-operation is invoked if the angle is larger than $\frac{\pi}{2}$, otherwise the shorten-operation is used.

Hop-operation: Let $\vec{a}$ a be the vector such that $p'_{i-1} + \vec{a} = p_i$. Then the hop-operation executed by relay r_i moves it to position $p'_i = p_{i+1} - \vec{a}$. The described operation is equivalent to mirroring r_i's position with respect to the midpoint of the line through p'_{i-1} and p_{i+1}. After the hop-operation has been executed, the sequential run proceeds at the next relay.

Shorten-operation: In the shorten-operation the relay r_i moves to the middle of the line segment between p'_{i-1} and p_{i+1}, just as in the Go-To-The-Middle strategy. Executing the shorten-operation finishes the run of the Hopper strategy and thus r_{i+1} does not move in the next time step.

Theorem 4.3 ([27]). *Starting with a chain with n relays, the* Hopper *strategy ensures that after $8n+1$ runs, i.e., $25n+1$ time steps, the chain uses at most $2\sqrt{2}d + 1$ relays.*

The main idea is to show that hop runs that carry through the entire chain deliver progress. The progress measure is a monotone property that is linearly bounded. Since runs that do not carry through the chain are stopped either by a shorten or a remove operation, the remainder of the analysis concentrates on bounding the number of these operations. Note that the step length is slightly larger than for the Go-To-The-Middle-strategy in the worst case, but still upper bounded by $\sqrt{2}$ (for Go-To-The-Middle, the step-length is bounded by 1).

By now, we have seen an intuitive strategy for the Short Chain Problem with essentially quadratic runtime and a

more complex strategy with linear runtime. This gives a deeper insight into the problems considered in Section 3 from the local point of view. However, the differences in the models used for the GO-TO-THE-MIDDLE strategy and the HOPPER strategy demand for a more detailed analysis of the reasons for the different results. We will do so in the next section.

5. The Short Chain Problem under refined performance measurements

In the previous section we have seen that the HOPPER-strategy is faster than the GO-TO-THE-MIDDLE-strategy in terms of time steps. On the other hand, since the HOPPER covers longer distances in one step than GO-TO-THE-MIDDLE, the movement of HOPPER seems to be more imprecise: the robots might move larger distances in the wrong direction and thus the distance covered by robots using the HOPPER-strategy might be larger than when using GO-TO-THE-MIDDLE. In practice, there are two main energy consumers for mobile robots: measurements/communication and movement. As a robot performs only one Look operation per time step, the number of time steps is equal to the number of measurements, but our analysis in Section 4 does not take the energy into account which is spent for movement. When disregarding the measurements and only considering the distance traveled by the robots, it seems natural that more measurements lead to a better result. We will therefore not only analyze the GO-TO-THE-MIDDLE-strategy as presented in the last section, but we will also consider a variant of GO-TO-THE-MIDDLE in which the robots are limited to a step length of δ in each time step, $0 < \delta \leq 1$, where $\delta = 1$ corresponds to the original GO-TO-THE-MIDDLE-strategy. Besides measuring the traveled distance, the number of rounds is still of interest.

When using only the movement distance as a quality measure, an obvious idea is to reduce the step length further with $\delta \to 0$ to obtain a continuous time model in which a continuous observation of the environment is possible. It follows that the direction in which a relay moves can change continuously over time. In this model exists a natural strategy which is similar to GO-TO-THE-MIDDLE, where each relay r moves in direction of the bisector of the angle at r which is formed by r and its two neighbors. Note that this is a stronger model since the robots do not need to know the exact positions of their neighbors but only the corresponding angle, which is strictly less information. It turns out that the strategy is optimal up to constant factors.

5.1. *The δ-bounded model*

In the δ-bounded model the setting and the algorithm are exactly as for the GO-TO-THE-MIDDLE strategy. However, as described above the step length is bounded to δ. This leads to two phases during the execution of the algorithm. In the first phase, at least one robot cannot reach its target position, which was computed in the Compute-step, and stops after he has traveled a distance of δ. In the second phase, the target position can be reached by all robots. As soon as this is the case, the strategy is equivalent to the GO-TO-THE-MIDDLE strategy. Therefore the bounds in the following theorems consists of two summands, one for the first phase and one for the second phase. We now give the bounds for the needed number of time steps. The results from this section are published in [28].

Theorem 5.1 ([28]). *When applying the δ-bounded* GO-TO-THE-MIDDLE*-strategy, after $\mathcal{O}(n^2 \log n + \frac{n}{\delta})$ time steps for every r_i holds $d(p_i) \leq 1$, where $d(p_i)$ is the distance of relay r_i to its optimal final position. Moreover, there exists a configuration such that the number of time steps is $\Omega(n^2 + \frac{n}{\delta})$ until $d(p_i) \leq 1$ for all r_i.*

The lower bound of $\Omega(n^2)$ follows from the bound in Theorem 4.2, while the lower bound of $\Omega\left(\frac{n}{\delta}\right)$ also uses a triangular start configuration: one can bound the number of steps until the relay in the middle reaches the second phase. For the upper bound the $\mathcal{O}(n^2 \log n)$ bound follows from Theorem 4.1. The upper bound of $\mathcal{O}\left(\frac{n}{\delta}\right)$ can be shown by setting an upper bound on the maximum distance traveled using a more involved analysis.

As pointed out earlier, one of the main motivations to consider the δ-bounded model was to measure the traveled distance. Here is the main result.

Theorem 5.2 ([28]). *When applying the δ-bounded* GO-TO-THE-MIDDLE*-strategy, the maximum distance traveled by a relay is $\Theta(\delta n^2 + n)$ in the worst case.*

The first phase takes $\Theta\left(\frac{n}{\delta}\right)$ steps, which is shown as part of the proof of Theorem 5.1. Since there is a relay which always moves a distance of δ in the first phase, the maximum distance traveled by a relay is of order $\Theta(n)$. The second phase consists only of GO-TO-THE-MIDDLE steps. Different to the number of time steps, for the analysis of the traveled distance, $t \to \infty$ can be considered. The distance that can be traveled by a relay depends on the respective distance that the neighboring robots are moving. The terminals at the end of the chain do not move at all. Therefore the distance their neighbors can travel is bounded. This propagates through the chain, reaching the maximal possible traveling distance at the relay with index $\frac{n}{2}$. This maximal distance can be computed and leads to the stated bound.

An important special case is $\delta \to 0$, since this resembles the GO-TO-THE-MIDDLE strategy when no energy for measurements is taken into account.

Theorem 5.3 ([28]). *When the* GO-TO-THE-MIDDLE *strategy for $\delta \to 0$ is performed, the maximum distance traveled by a relay is $\Theta(n)$ for a worst-case start configuration.*

This is a consequence of the fact that there is only the first phase. However, the target points calculated in this model are never reached by the relays. Therefore, a natural question is whether less information is required to obtain similar results. We will consider this next.

5.2. *The* MOVE-ON-BISECTOR*-strategy*

In the last subsection we reduced the step length of the GO-TO-THE-MIDDLE-strategy. This was motivated by the fact that movement as well as measurements cost energy. We also considered the GO-TO-THE-MIDDLE-strategy in a model where measurements were for free and only movement is

costly by letting $\delta \to 0$ (the *continuous* model). In this model, there is a more natural strategy which needs less robot abilities: the MOVE-ON-BISECTOR-strategy. Since movement is continuous, relays do not need to compute a target position in the Compute-Step, but only a direction in which to move. The MOVE-ON-BISECTOR-strategy now works as follows. At every point of time, every relay r_i computes the internal angle of the triangle p_{i-1}, p_i, p_{i+1} at p_i. It then chooses the bisector of this angle as the direction in which to move. If r_i has already reached the line segment between its neighbors and thus the internal angle is 180 degrees, it stays on this line segment. Note that there is no delay when the direction in which to move is computed. Note further that for this strategy, relays do not need the ability to determine the positions of their neighbors: they only need to know the direction and not the distance in which they are positioned. This ability is by far easier to implement in reality than position measurements. We will now investigate the maximum distance traveled by a relay as in the last subsection. The results were published in [29].

Theorem 5.4 ([29]). *When the* MOVE-ON-BISECTOR*-strategy is performed in the continuous model, the maximum distance traveled by a relay is $\Theta(n)$ for a worst-case initial configuration.*

The proof of the upper bound follows from a more general theorem, where the traveled distance is investigated in dependency on the initial configuration in terms of the distance from the final configuration and the length of the chain. The lower bound is obvious, since there are configurations where a relay has to travel a distance of $\Omega(n)$ to reach the line between the terminals even with an optimal algorithm.

We have analyzed the problem of locally forming communication infrastructures for the important special case of forming lines. Here, we focused on the *formation* part and provided therefore structural information by giving the neighborhood relationships. In the next chapter, we will study changing neighborhoods, but drop the requirement of forming real formations. Therefore, we consider the Gathering Problem.

6. The Gathering Problem

Like the Short Chain Problem, the Gathering Problem is a simplification of the Sparse Communication Network Problem. We have again a group of n relays (robots) in the plane, but this time no terminals are used. The goal is to gather all robots in one point. In comparison to the problems studied in Sections 3–5, we do not need to form short communication structures, but focus on the two-dimensional properties of the original problem. Like in the Sparse Communication Network Problem, neighbors of a robot are not predetermined, but they are defined by a local viewing range. Thus, with the Gathering Problem we capture the challenge of changing neighborhoods.

We again want to focus on the local aspects of this problem. So we use again robots with a local viewing range which are oblivious and do not exchange information in the network. The GO-TO-THE-CENTER-strategy can be used in this setting, too, except for the deletion of robots. Omitting this rule, the following theorem follows immediately from [22].

Theorem 6.1 ([22]). *The* GO-TO-THE-CENTER*-strategy without deletion gathers the robots in one point.*

A very similar strategy for the Gathering Problem is investigated in [12]. The authors also take the center of the smallest enclosing circle as the target position for the robots, but they use the synchronous time model instead of the asynchronous one. To avoid the loss of connectivity, they restrict the step length of robot r in one time step, which is dependent on the positions of the robots in r's viewing range. The authors show that this variant still gathers the robots.

Theorem 6.2 ([12]). *The synchronized* GO-TO-THE-CENTER*-strategy gathers the robots in one point.*

No runtime bounds are so far known for this restricted robot and locality model, but we now present a more complex strategy for slightly more powerful robots which achieves gathering in expected $\mathcal{O}(n^2)$ rounds.

The MOVE-IN-CH*-strategy.* With MOVE-IN-CH (Move in Convex Hull), we present a strategy which works in the same local environment as GO-TO-THE-CENTER. Thus, robots have a limited viewing range, are oblivious and do not exchange information. However, robots are not only able to move themselves, but also robots from their vicinity. This ability allows the robots to perform *fusions*, that is, robots move to the same position and stay together from this point in time on. Fusing two robots is therefore equivalent to deleting one of them. Moreover, we assume a viewing range of 2, whereas the communication range is 1. We now demand to keep the unit disk graph with respect to the communication range connected at all times.

The MOVE-IN-CH-strategy is performed in asynchronous rounds, but our analysis further assumes a randomized activation order. There are several possibilities for how to use randomization. The first possibility is to choose the next active robot uniformly at random from all robots, or one can choose the activation sequence for the next round uniformly at random from all permutations of robots. A local variant is to let each robot throw a coin to decide whether it turns to a ready state and then letting those robots turn active which are the only ready robots in their neighborhood.

When a robot r turns active, it performs the algorithm described in Table 1.

See Fig. 2 for an illustration of Step 3b of the algorithm. Note that the algorithm is deterministic. We can bound the expected value for the number of rounds until all robots have gathered in one point; the only randomness used is the stochastic round model. In particular, the algorithm can also be executed in an asynchronous worst case round model, the only difference is that we cannot guarantee the runtime in this case.

Analysis of MOVE-IN-CH. We will now bound the expected number of rounds until gathering is achieved. This analysis covers the proof that the algorithm is correct and gathering is always achieved.

Theorem 6.3 ([30]). *The* MOVE-IN-CH*-strategy needs expected $O(n^2)$ rounds to gather the robots in one point.*

For the runtime analysis, we use two progress measures. In each round, either two robots are fused or the area of

Table 1 – Algorithm MOVE-IN-CH.

MOVEINCH: The algorithm for robot r at time t:

1. Compute the sets A_r and B_r of the robots within the viewing resp. communication range of r. Let C_r denote the convex hull of A_r.
2. (Termination) If $A_r = B_r$ (i.e. no robots from A_r have distance between 1 and 2 to r), then move all robots from A_r to the position p_r of r.
3. Else (B_r is a proper subset of A_r)

3.a (Fusion) If the positions in B_r can be rearranged such that the resulting new set A'_r is still contained in C_r, is still connected, and at least two robots share the same position (are fused), perform this rearrangement. Fused robots will alway have the same position from now on.

3.b (Reduction) If fusion is not possible and r lies on the boundary of C_r, they do the following:

(a) Compute the two first intersections of the boundary of C_r with the boundary of r's communication range if started from p_r on C_r in clockwise/counterclockwise direction. (Note that these are the intersections which are in maximum distance to each other.)

(b) Compute the line segment l between these intersections.

(c) Move all robots on r's side of l to their respective closest point on l.

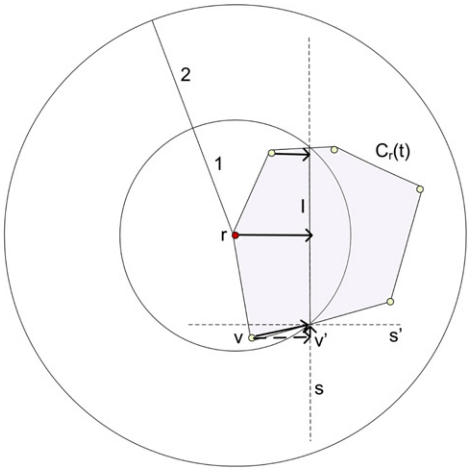

Fig. 2 – Illustration of step 3b of the algorithm and its correctness.

the convex hull of the robots is reduced in expectation by a constant. Since there are only n robots, fusions can occur in at most $n - 1$ rounds. Moreover, because the unit disk graph of the robots is connected at the beginning, the area of the convex hull of the robots is at most $\mathcal{O}(n^2)$. Robots never leave the current convex hull, and thus the number of rounds without fusions is in expectation $\mathcal{O}(n^2)$. Theorem 6.3 follows.

7. Conclusion and outlook

We have presented approximation algorithms for the general Sparse Communication Network Problem and pointed out that finding a local and distributed algorithm for this problem is difficult due to cycles in the unit disk graph which cannot be detected by robots with a local viewing range. The open question which follows is: how must we strengthen the robots such that a good network can be reached? How can we detect circles and nevertheless maintain some locality?

The approach taken was to restrict the problem in two ways. First, each relay got the additional information of two neighbors and we presented several algorithms for using this information to build a straight line between two terminals. This problem can possibly be relaxed by still letting the relays form the line between the terminals, but not giving them information about neighbors. We have seen in Section 3 that the GO-TO-THE-CENTER-strategy builds such a line, but runtime bounds for this problem remain open.

The second restriction was to omit the search for a structure, but to let the relays gather in one point. The open questions here are whether gathering can also be reached quickly with weaker robots. In particular, is it possible to use a viewing range of 1 instead of 2? Is it necessary that relays are able to move their neighbors? Another research direction here is to adapt the refined analysis for the relay chain problem in Section 5. In particular, what happens if we restrict the step length to δ? And can we find an algorithm which works in continuous time?

Acknowledgements

This work has been 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.

REFERENCES

[1] Errol L. Lloyd, Guoliang Xue, Relay node placement in wireless sensor networks, IEEE Transactions on Computers 56 (1) (2007) 134–138.

[2] Anand Srinivas, Gil Zussman, Eytan Modiano, Mobile backbone networks—construction and maintenance, in: Proc. 7th Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2006, ACM Press, 2006, pp. 166–177.

[3] Alon Efrat, Sándor P. Fekete, Poornananda R. Gaddehosur, Joseph S.B. Mitchell, Valentin Polishchuk, Jukka Suomela, Improved approximation algorithms for relay placement, in: Proc. 16th European Symposium on Algorithms, ESA 2008, 2008, pp. 356–367.

[4] X. Défago, A. Konagaya, Circle formation for oblivious anonymous mobile robots with no common sense of orientation, in: International Workshop on Principles of Mobile Computing, POMC, 2002, pp. 97–104.

[5] I. Chatzigiannakis, M. Markou, S. Nikoletseas, Distributed circle formation for anonymous oblivious robots, in: Workshop on Efficient and Experimental Algorithms, WEA, 2004, pp. 159–174.

[6] Yoann Dieudonné, Franck Petit, Self-stabilizing deterministic gathering, in: Algorithmic Aspects of Wireless Sensor Networks, 2009, pp. 230–241.

[7] Samia Souissi, Xavier Défago, Masafumi Yamashita, Gathering asynchronous mobile robots with inaccurate compasses, in: Principles of Distributed Systems, 2006, pp. 333–349.

[8] Taisuke Izumi, Yoshiaki Katayama, Nobuhiro Inuzuka, Koichi Wada, Gathering autonomous mobile robots with dynamic compasses: an optimal result, Distributed Computing (2007) 298–312.
[9] Giuseppe Prencipe, Impossibility of gathering by a set of autonomous mobile robots, Theoretical Computer Science 384 (2–3) (2007) 222–231. Structural Information and Communication Complexity (SIROCCO 2005).
[10] Noa Agmon, David Peleg, Fault-tolerant gathering algorithms for autonomous mobile robots, in: SODA'04: Proceedings of the Fifteenth Annual ACM–SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2004, pp. 1070–1078.
[11] Reuven Cohen, David Peleg, Convergence properties of the gravitational algorithm in asynchronous robot systems, SIAM Journal on Computing 34 (6) (2005) 1516–1528.
[12] Hideki Ando, Yoshinobu Suzuki, Masafumi Yamashita, Formation agreement problems for synchronous mobile robots with limited visibility, in: Proc. IEEE Syp. of Intelligent Control, 1995, pp. 453–460.
[13] Hideki Ando, Yoshinobu Oasa, Ichiro Suzuki, Masafumi Yamashita, Distributed memoryless point convergence algorithm for mobile robots with limited visibility, IEEE Transactions on Robotics and Automation 15 (5) (1999) 818–828.
[14] Donghui Chen, Ding-Zhu Du, Xiao-Dong Hu, Guo-Hui Lin, Lusheng Wang, Guoliang Xue, Approximations for Steiner trees with minimum number of Steiner points, Journal of Global Optimization 18 (1) (2000) 17–33.
[15] Donghui Chen, Ding-Zhu Du, Xiao-Dong Hu, Guo-Hui Lin, Lusheng Wang, Guoliang Xue, Approximations for Steiner trees with minimum number of Steiner points, Theoretical Computer Science 262 (1–2) (2001) 83–99.
[16] Hai Liu, Pengjun Wan, Xiaohua Jia, On optimal placement of relay nodes for reliable connectivity in wireless sensor networks, Journal of Combinatorial Optimization 11 (2006) 249–260.
[17] Weiyi Zhang, Guoliang Xue, Satyajayant Misra, Fault-tolerant relay node placement in wireless sensor networks: problems and algorithms, in: Proc. 26th Conference on Computer Communications, INFOCOM 2007, 2007, pp. 1649–1657.
[18] Jonathan L. Bredin, Erik D. Demaine, Mohammad Taghi Hajiaghayi, Daniela Rus, Deploying sensor networks with guaranteed capacity and fault tolerance, in: Proc. 6th Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2005, ACM Press, 2005, pp. 309–319.
[19] Yang Yang, Mingen Lin, Jinhui Xu, Yulai Xie, Minimum spanning tree with neighborhoods, in: Proc. 3rd Conference on Algorithmic Aspects in Information and Management, AAIM 2007, 2007, pp. 306–316.
[20] Ding-Zhu Du, Frank K. Hwang, An approach for proving lower bounds: solution of Gilbert–Pollak's conjecture on Steiner ratio, in: Proc. 31st Symposium on Foundations of Computer Science, FOCS 1990, 1990, pp. 76–85.
[21] Joseph S.B. Mitchell, Guillotine subdivisions approximate polygonal subdivisions: a simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems, SIAM Journal on Computing 28 (4) (1999) 1298–1309.
[22] Friedhelm Meyer auf der Heide, Barbara Schneider, Local strategies for connecting stations by small robotic networks, in: IFIP International Federation for Information Processing, in: Biologically—Inspired Collaborative Computing, vol. 268, Springer, Boston, 2008, pp. 95–104.
[23] Barbara Schneider, Lokale strategien zur aufrechterhaltung von kommunikation zwischen mobilen robotern, Master's Thesis, University of Paderborn, 2008.
[24] Miroslaw Dynia, Jarosław Kutyłowski, Paweł Lorek, Friedhelm Meyer auf der Heide, Maintaining Communication Between an Explorer and a Base Station, in: IFIP International Federation for Information Processing, vol. 216, Springer, Boston, 2006, pp. 137–146.
[25] Miroslaw Dynia, Jarosław Kutyłowski, Friedhelm Meyer auf der Heide, Jonas Schrieb, Local strategies for maintaining a chain of relay stations between an explorer and a base station, in: SPAA'07: Proceedings of the Nineteenth Annual ACM Symposium on Parallel Algorithms and Architectures, ACM Press, 2007, pp. 260–269.
[26] Jaroslaw Kutylowski, Using mobile relays for ensuring connectivity in sparse networks, Dissertation, International Graduate School of Dynamic Intelligent Systems, December 2007.
[27] Jarosław Kutyłowski, Friedhelm Meyer auf der Heide, Optimal strategies for maintaining a chain of relays between an explorer and a base camp, Theoretical Computer Science 410 (36) (2009) 3391–3405.
[28] Philipp Brandes, Bastian Degener, Barbara Kempkes, Friedhelm Meyer auf der Heide, Building short chains of mobile robots locally with a bounded stepwidth, Preprint, 2010. wwwhni.uni-paderborn.de/alg/publikationen.
[29] Bastian Degener, Barbara Kempkes, Peter Kling, Friedhelm Meyer auf der Heide, A continuous, local strategy for constructing a short chain of mobile robots, in: 17th International Colloquium on Structural Information and Communication Complexity, 2010.
[30] Bastian Degener, Barbara Kempkes, Friedhelm Meyer auf der Heide, A local $O(n^2)$ gathering algorithm, in: SPAA'10: Proceedings of the 22nd ACM Symposium on Parallelism in Algorithms and Architectures, June 2010.

Bastian Degener, born in 1979, received his Ph.D. from the Department of Computer Science of the University of Paderborn in 2010. He is currently working in the group Algorithms and Complexity at the University of Paderborn. His research interests include distributed and parallel computing in local and dynamic scenarios as well as wireless sensor networks.

Sándor P. Fekete received his Doctoral Degree in Combinatorics and Optimization from the University of Waterloo Science, Canada (1992). Currently, he is full professor for Algorithmics in the Department of Computer Science of the Braunschweig University of Technology, where he is also Department Chair. His research interests lie in both theoretical and practical aspects of algorithms, with a wide variety of interdisciplinary collaborations. One of his main interests has been distributed algorithms, in particular in the context of large sensor networks. He has coauthored about 150 refereed scientific publications.

Barbara Kempkes, born in 1983, obtained her Diploma in Business Computing from the Faculty of Business Administration and Economics of the University of Paderborn in 2008. She is currently a Ph.D. student at the Computer Science Department of the University of Paderborn (since May 2008) under the supervision of Prof. Dr. Friedhelm Meyer auf der Heide. Her research interests include distributed and parallel computing in local and dynamic scenarios as well as wireless sensor networks.

Friedhelm Meyer auf der Heide received his Ph.D. in Mathematics at the University of Bielefeld, Germany, in 1981. He is currently Full Professor for Algorithms and Complexity at the Heinz Nixdorf Institute and at the Computer Science Department of the University of Paderborn, Germany. He was coordinator of the European Integrated Project: Dynamically Evolving, Large Scale Information Systems (DELIS) and several German coordinated research programs in Paderborn. His research interests include algorithmic and complexity theoretical problems concerning parallel computing, communication and data management in networks, dynamics in networks, algorithms in computer graphics, and probabilistic analysis. He is (co-)author of more than 140 publications and several patents.