A distributed minimum restrictive connectivity maintenance algorithm

Jay Wagenpfeil * **Adrian Trachte** * **Takeshi Hatanaka** **
Masayuki Fujita ** **Oliver Sawodny** *

* *Institute for System Dynamics, University Stuttgart, 70569 Stuttgart, Germany, e-mail: jay.wagenpfeil@isys.uni-stuttgart.de*
** *Department of Mechanical and Control Engineering, Tokyo Institute of Technology, Tokyo 152-8552, Japan*

Abstract: The dynamic character of autonomous mobile multi-agent networks makes the control of the individual agent's movement a challenging task when certain properties or conditions concerning the global network state must be fulfilled. Many distributed algorithms require that the communication topology of the regarded multi-agent network is connected at all times to execute properly. Since particularly motion control algorithms directly influence the connectivity itself, the fundamental question how connectivity of the network can be maintained is raised. As an additional difficulty, individual agents often have only access to local information, whereas connectivity is a property of the global network. In this work, a distributed hybrid connectivity maintenance algorithm is proposed, which guarantees global connectivity of the network based on local information. The algorithm is divided in two components: First, a discrete topology control algorithm determines periodically the structure of a minimum cost spanning tree on the network communication graph. Second, a continuous connectivity motion control algorithm constrains the location of each individual agent such that the determined spanning tree is preserved. This combined algorithm effectively maintains connectivity under the presence of any finite-valued primary motion control law.

Keywords: connectivity maintenance, distributed algorithm, local information, dynamic shortest path search, multi-agent networks, motion coordination

1. INTRODUCTION

Over the last ten years, the analysis and control of multi-agent networks has attracted the attention of a growing number of researchers, since their relevance to applications like formation control of autonomous ground and aerial vehicles, highway traffic control or modeling of social networks. Particularly well studied is the cooperative motion control of mobile robotic agents. Another topic that has drawn much attention is distributed decision making. There exist many more distributed algorithms for multi-agent networks. As a common property, most algorithms require communication between individual agents. Due to energy limitations, interferences and other disturbances of the communication medium it is generally not possible for an agent to communicate with every other agent in the network. To ensure a complete information exchange within the network, it is usually assumed that agents can relay messages between neighbors. By that a multi-hop communication network forms which provides the information exchange within the network. It is, however, necessary that the topology of this multi-hop communication network satisfies certain properties, particularly it is required that the network is connected.

Guaranteeing connectivity based on local information of the network is difficult, since it is a global task. Moreover, connectivity of the network is directly influenced by the movement of the agents due to a cooperative motion control law. Therefore it is often assumed, that connectivity is just given. This limits the use and practical applicability of many algorithms. To overcome this problem, a new hybrid connectivity maintenance control algorithm is proposed in this work. It combines a discrete topology control algorithm with an continuous motion control algorithm and can be used to guarantee connectivity in various multi-agent network scenarios, including motion coordination tasks with arbitrary finite values motion controllers.

The rest of this work is organized as follows. In section 2 a short survey of related work and previous research in the context of the here presented work. The problem formulation is stated in section 3 together with some fundamental notions and definitions that are necessary for the analysis. In section 4 a connectivity maintenance algorithm is stated and its properties are analyzed. The simulation presented in section 5 will help illustrating the effectiveness of the proposed algorithm. In section 6 a concluding analysis and directions for future research are given.

2. RELATED WORK

Early research focused on the connectivity of static networks, mainly under the aspect of power consumption, see for example the work by Wang et al. (2003) or Chen et al. (2002). To estimate the connectedness of a multi-agent network only based on local information, relations between

the connectivity and the minimum node degree – that is the minimal number of each agent's neighbors – have been derived, see for instance the work by Bettstetter (2002) or Xue and Kumar (2004). More recent research deals with the connectivity of mobile network under particular motion control tasks. Poduri and Sukhatme (2004) combine two virtual forces that repel or attract neighboring agents based on the number of each agent's neighbors. This results in a coverage behavior of the network. However, since this method also assumes that connectivity is preserved by maintaining a minimum node degree, connectivity cannot be guaranteed as the network evolves. Srivastava and Spong (2008) and Zavlanos and Pappas (2005) develop two similar, centralized algorithms that ensure connectivity of the network under the presence of an arbitrary motion control law. This is done by constraining the movement of each agent, such that the connection to a particular subset of its neighbors is preserved. However, such centralized algorithms contradict the idea of distributed networks and cannot be performed based on only local information.

A decentralized algorithm that guarantees connectivity is presented by Bullo et al. (2008), that simply constrains each agent's movement such that the connection to all its neighbors is preserved. This is obviously a rather restrictive approach that might result in highly conservative results. Spanos and Murray (2004) introduce a connectivity robustness function which rates the importance of each agent's neighbors. The gradient of this function is used to maintain the connectivity of the network. However, this connectivity robustness function must be parametrized depending on the given problem setup and motion control law, which limits its practical use. Zavlanos and Pappas (2007) also developed a distributed connectivity control algorithm, which is based on a local estimation of the network topology. Each agent stores in a routing table the connections to direct neighbors that must be preserved in order to maintain connectivity. Since only one agent's routing table can be updated at a time, the agents have to run a sophisticated coordination protocol to decide which agent can update its routing informations. All of the mentioned algorithms for mobile networks share the drawback that they generally preserve more connections between the agents than necessary to guarantee connectivity. In consequence, the movement of the agents is more conservatively restricted than required, which might result in a reduced performance or even failure of the given motion task.

The approach presented in this work reduces the restrictions imposed by the connectivity maintenance control to the necessary minimum. This is done by calculating only the absolutely necessary links that must be preserved, based on a shortest path search with a particular virtual cost function. Moreover, this topology control algorithm dynamically updates the links that must be preserved such that the network topology can flexibly adjust to the movement of the network.

3. FUNDAMENTAL NOTIONS

In this section, the problem setup and some fundamental notions necessary for the analysis are stated. Consider a network of n autonomous agents that are located in the d-dimensional space and let $s_i(t) \in \mathbb{R}^d$ be the location of the i^{th} agent at time t. Moreover, let $s(t) = [s_1(t), \ldots, s_n(t)]^{\mathrm{T}} \in \mathbb{R}^{nd}$ be the stacked positions of all agents. The location of an agent i satisfies the first order dynamics $\dot{s}_i(t) = u_i(t)$ with the input u_i. It is assumed that an inner control loop encapsulates the generally more complex dynamics of the agent's physical model such that the agent may move according to the given input u_i. The primary objective of the agent is to follow a movement given by an arbitrary, finite valued motion control law $u_{\mathrm{mc},i}$. A connectivity controller overlays the input generated by the motion controller with an connectivity input $u_{\mathrm{cc},i}$ that ensures that agent i does not move out of communication range with certain critical neighbors. The resulting input is hence given by $u_i = u_{\mathrm{mc},1} + u_{\mathrm{cc},i}$. The critical neighbors are determined by the topology controller based on a distributed algorithm that generates a minimum cost spanning tree. For this purpose, each agent i is equipped with a communication device and that allows a bi-directional communication with any agent j if $\|s_i(t) - s_j(t)\| \leq r_{\mathrm{com}}$, where r_{com} is the so called communication range and $\|\cdot\|$ is the Euclidean norm. The structure of the connectivity maintenance control algorithm within the structure of the agent is shown in figure 1. Note further, that throughout this work it is assumed that the network is initially connected.

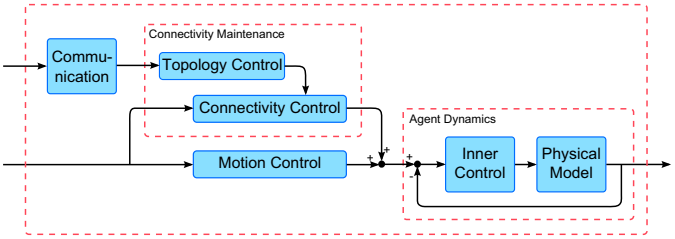

Fig. 1. Agent model decomposition.

3.1 Graphs

The topology control is based on shortest paths on dynamic graphs, therefore a brief introduction to algebraic graph theory will be given now. A *graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of order n is given by a set of n *vertices* $\mathcal{V} = \{v_1, \ldots, v_n\}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. A graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ is a *subgraph* of the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$. In an *undirected* graph, for each pair of vertices $(u, v) \in \mathcal{E}$ holds $(v, u) \in \mathcal{E}$. The neighbors of a vertex v are given by the set $\mathcal{N}(v) = \{u \in \mathcal{V} \,|\, (u, v) \in \mathcal{E}\}$. The *degree* $d(v)$ of a vertex v is given by the number of neighbors, i.e. the cardinality of $\mathcal{N}(v)$. A *path* l on the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a sequence of m (possibly repeating) edges, such that any pair of consecutive edges shares one common vertex. All paths between u and v are contained in the set $\mathcal{L}(u, v)$. Two vertices are said to be *connected*, if there exists a path between those vertices. A graph is *connected* if any pair of vertices is connected. A nontrivial path that starts and ends at the same vertex is called a *cycle*. A graph is *acyclic* if it does not contain any cycles. A connected acyclic graph is a *tree*. The graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ contains a so called *spanning tree* $\mathcal{G}_{\mathrm{ST}} \subseteq \mathcal{G}$, that is a tree that connects all vertices $\mathcal{V}$ of the graph, if and only if $\mathcal{G}$ is connected.

A *weighted graph* is given by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ where $(\mathcal{V}, \mathcal{E})$ is an undirected graph and $\mathcal{A}(v_i, v_j)$ is a *weighting function* $\mathcal{A} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}_0^+$ with the property that $\mathcal{A}(v_i, v_j) > 0$ if

$(v_i, v_j) \in \mathcal{E}$ and $\mathcal{A}(v_i, v_j) = \infty$ otherwise.[1] The topology of a weighted graph is uniquely determined by the set of vertices $\mathcal{P}$ therefor the weighting function $\mathcal{A}$ and the edge set $\mathcal{E}$ can be omitted. The *weighted distance* from a vertex u to v is the smallest accumulated weight along any path $l \in \mathcal{L}(u, v)$ from u to v and ∞ if $\mathcal{L}(u, v) = \emptyset$. The path corresponding to the weighted distance is the so called *shortest path* l^*. A minimum cost spanning tree $\mathcal{G}^*_{\mathrm{ST}}$ is a tree such that every pair of vertices (u, v) is connected along the shortest path between u and v. A useful tool to describe a dynamically changing network topology is the so called *proximity graph*. A proximity graph $\mathcal{G}(s, \mathcal{A})$ of a multi-agent network is a graph, whose topology depends on the locations s of the agents and the weighting function $\mathcal{A} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+_0$. Given a multi-agent network with the previously stated property that two agents i and j can communicate if $\|s_i - s_j\| \leq r_{\mathrm{com}}$, then this communication topology can be described by the r-disk proximity graph $\mathcal{G}_{\mathrm{disk}}(s(t), r_{\mathrm{com}})$ whose edge-defining weighting function $\mathcal{A}$ satisfies $\mathcal{A}(s_i, s_j) < \infty$ if $\|s_i - s_j\| \leq r_{\mathrm{com}}$ and $\mathcal{A}(s_i, s_j) = \infty$ otherwise.

4. CONNECTIVITY MAINTENANCE ALGORITHM

The connectivity maintenance algorithm is separated into two distinct control tasks – the topology control and the actual connectivity control input generation. Both tasks make use of an artificial potential function $f : \mathbb{R}^d \to \mathbb{R}^+_0$. The topology controller runs a distributed algorithm to compute a minimum cost spanning tree on the communication graph, based on a virtual communication cost, defined by f depending on the distance over which needs to be communicated. The connectivity controller generates an input that will move the agent such that connectivity to critical neighbors – namely the agents that are neighbors on the minimum cost spanning tree – is maintained. The input value is computed based on the gradient of the artificial potential function. To ensure that the connectivity maintenance input dominates any finite valued motion control law $u_{\mathrm{mc},i}$, the potential function $f(p)$ with $p \in \mathbb{R}^d$ and $\|p\| \leq r_{\mathrm{com}}$ must satisfy the following properties:

- $f(p)$ is continuously differentiable,
- $f(p)$ is monotonically increasing with $\|p\|$,
- $f(0) = 0$, and
- $\lim_{\|p\| \to r_{\mathrm{com}}} f(p) = \infty$.

To minimize the influence of the connectivity maintenance input, it is further desired that the gradient of the function is close to zero for $\|p\| \ll r_{\mathrm{com}}$. A feasible potential function that satisfies these properties is given by $f(p) = \|p\| \cdot (r_{\mathrm{com}} - \|p\|)^{-m}$ with $m \gg 1$.

4.1 Topology control algorithm

Consider a multi-agent network with the previously stated property that two agents i and j can communicate if $\|s_i - s_j\| \leq r_{\mathrm{com}}$, and a communication cost $f : \mathbb{R}^d \to \mathbb{R}^+_0$ that associates a cost to the distance $d_i = \|s_i - s_j\|$ over which is communicated. The resulting weighted communication topology is described by the communication graph $\mathcal{G}_{\mathrm{com}}(s)$, which is given by the r-disk proximity graph with the modified weighting function $\mathcal{A}_f$ with the property that $\mathcal{A}_f(s_i, s_j) = f(s_i - s_j)$ if $\|s_i - s_j\| \leq r_{\mathrm{com}}$ and $\mathcal{A}(s_i, s_j) = \infty$ otherwise. Assuming that the network is connected at time t_0, then $\mathcal{G}_{\mathrm{com}}(s(t_0))$ contains at least one spanning tree $\mathcal{G}_{\mathrm{ST}} \subseteq \mathcal{G}_{\mathrm{com}}(s(t_0))$. Consequently, if the communication graph always contains a spanning tree as the network evolves for $t \geq t_0$, then the network remains connected. A spanning tree of particular interest is the so called minimum cost spanning tree $\mathcal{G}^*_{\mathrm{ST}}$. It can be easily obtained by defining one vertex of the communication graph as the root node, and computing then the shortest paths from all other vertices to the root. The union of the shortest paths results in the minimum cost spanning tree. A well known way to compute the shortest path to a single node is given by the distributed Bellman-Ford algorithm. This algorithm is distributed in a way that it can be performed at each node of the graph and relies only on information about the direct neighbors of the node. Although the distributed Bellman-Ford algorithm is well studied (see for example *Distributed Algorithms* by Lynch (1996)), its functionality will briefly be explained to point out the particular difficulties when searching the shortest path on dynamic graphs. It is assumed that the topology control algorithm is executed synchronously by all agents at discrete time steps with interval Δt. For an in depth analysis of the properties of the here presented topology control and the case of asynchronous networks, please refer to the diploma thesis of Trachte (2008).

The distributed Bellman-Ford algorithm is performed iteratively. In every iteration, each node i asks each of its neighbors $j \in \mathcal{N}_i$ on the communication graph $\mathcal{G}_{\mathrm{com}}$ for the estimated cost c_j along the shortest path to the root node, and determines the cost $f_{ij}(\|s_i - s_j\|)$ for communicating with this neighbor. The new estimated cost along the shortest path from node i to the root is then given by $c_i = \min_j(c_j + f_{ij})$ and the neighbor $h_i := \operatorname{argmin}_j(c_j + f_{ij})$ is called the *downstream neighbor* of node i. Let, without loss of generality, node 1 be the root, then the initial costs before starting the algorithm are $c_1 = 0$ and $c_i = \infty$ for all $i = \{2, \ldots, n\}$. As the algorithm performs, the estimated costs c_i converge to the optimal values c^*_i and h^*_i becomes the true next node on the shortest path towards the root. The edges of the minimum spanning tree $\mathcal{G}_{\mathrm{ST}}$ are given by $\bigcup_{i=2}^{n} \{(i, h_i)\}$, that is the union of the edges built by each node and its downstream neighbor. The algorithm convergences after at maximum $(n-1)$ iterations.

The problems of the distributed Bellman-Ford algorithm on graphs with dynamically changing weighting and topology basically arise, because the information about the shortest path is iteratively propagated through the network. If the communication costs change during this process, out-dated information is propagated which might prevent the convergence of the algorithm or result in an incorrect assumption on the shortest path. These problems are circumvented by periodically restarting the algorithm and fixing the communication costs at the beginning of each period. Using the fixed communication costs f'_{ij}, the algorithm converges after $(n-1)$ iterations, as in the static case. Restarting the shortest path search is controlled by the root node. At the beginning of a new period, it sends a message to all its neighbors in the communication graph,

[1] Note that in standard literature usually $\mathcal{A}(v_i, v_j) = 0$ if $(v_i, v_j) \notin \mathcal{E}$. This modification is necessary for the later introduced shortest path search algorithm.

```
begin iteration
    if RESTART message received then
        save fixed communication costs f'_ij
        send RESTART message to neighboring agents
        wait Δt
        compute average communication costs f̄_ij
    end
    update communication cost to root
    update candidate downstream neighbour
end
```

Fig. 2. Topology control algorithm in pseudo code.

that tells those neighbors to fix the communication costs f and begin a new searching period. Each agent j that receives this restart message relays the message to all its neighbors $\mathcal{N}_j$. By that the information to restart the shortest path search is propagated through the network. Since the propagation of this message is at least as fast as the propagation of the shortest path, the convergence of the shortest path search algorithm is not changed. Note that during the search, only a candidate downstream neighbor is determined and the old downstream neighbor is left unchanged. After n iterations (that is before starting a new search period), the newly found candidate downstream neighbor is made the actual downstream neighbor.

There are, however, a few details that must be dealt with. First, it is necessary that $f'_{ij} = f'_{ji}$, which means that the fixed communication cost between two agents must be the same. Since it might happen that both agents fix the communication costs at different times, each agent i must compute the average fixed communication costs $\bar{f}_{ij} = 0.5(f'_{ij} + f'_{ji})$ in the step after it received the RESTART message. This requires one extra step and extends the convergence of the algorithm and thus the search period to n iterations. Second, it must be ensured that the shortest paths determined by the algorithm still exist at the end of the iteration period. Suppose two neighboring agents i and j such that $0 < r_{\text{com}} - \|s_i - s_j\| \ll 1$, which means the distance between the two is close to the communication range. Now assume that both agent move in opposite directions, such that at the end of the period $\|s_i - s_j\| > r_{\text{com}}$ holds. If now either of the agent determined that the shortest path leads along the other agent, then the algorithm breaks because the communication between these two agents is not possible anymore. It is therefore necessary to restrict the shortest path search to a subset of the graph that will not change its topology within the next search period. Suppose that $\Delta s_{\max}$ is the maximum distance that any agent can cover during one iteration step. Then, given an agent i and its neighbors $\mathcal{N}_i$ at the beginning of a search period, all agents $j \in \mathcal{N}_i$ for which $\|s_j - s_i\| < r_{\text{com}} - n\Delta s_{\max}$ holds will still be neighbors of agent i at the end of the period. When none of agent i's neighbors satisfies this condition, agent i's shortest path is not changed during the search period. Clearly, if $n * \Delta s_{\max} > r_{\text{com}}$ then no minimum cost spanning tree can be computed and the initial spanning tree is preserved. Therefore the computation speed must be sufficiently high, such that $\Delta s_{\max}$ is small respectively $n\Delta s_{\max} \ll r_{\text{com}}$. The resulting operational sequence of the topology control algorithm is shown as pseudo code in figure 2.

The topology control algorithm provides each agent i with the required knowledge of the local structure of the minimum cost spanning tree, in particular with the information on the downstream neighbor h_i, that is the next agent along the shortest path to the root. By a slight extension to the algorithm it is possible to provide agent i also with the knowledge, which agents consider i as their downstream neighbor. These agents are denoted as the *upstream neighbors* $\mathcal{U}_i \subseteq \mathcal{N}_i$ of agent i and given by $\mathcal{U}_i = \{j \in \mathcal{N}_i : i = h_j\}$. The set $\mathcal{N}_i^* = \mathcal{U}_i \cup \{h_i\}$ denotes agent i's *critical neighbors* which are its neighbors on the minimum cost spanning tree.

4.2 Connectivity control

The task of the connectivity controller is to constrain the location of the agent i such that the connection to the critical neighbors is preserved. In other words, the controller has to overlay the input generated by the motion controller $u_{\text{mc},i}$ such that the condition $\|s_i - s_j\| < r_{\text{com}}$ is satisfied for all $j \in \mathcal{N}_i^*$. If the connection to the critical neighbors is preserved, the topology of the graph and particularly is connectivity is maintained.

Before looking at the actual connectivity control algorithm, suppose a preliminary controller $\tilde{u}_{\text{cc},i}$ that preserves only the connection to agent i's downstream neighbor h_i. Clearly, if each agent in the network keeps connected with its downstream neighbor, then connectivity of the complete network is maintained.

Lemma 1. Let f be a potential function satisfying the above stated properties. Then the connectivity control law

$$\tilde{u}_{\text{cc},i} = -\left.\frac{\partial f}{\partial s_i}\right|_{s_i - s_{h_i}}^{\mathrm{T}} \tag{1}$$

maintains the connectivity of agent i and its downstream neighbor h_i under any finite valued motion control law $u_{\text{mc},i}$.

Proof. For simplicity, let $\nabla f_i = \frac{\partial f}{\partial s_i}$ denote the partial derivate of the function with respect to agent i's location s_i. Using $V_i(s_i) = f(s_i - s_{h_i})$ as a candidate Lyapunov function, let $u_{\max} = \max |u_{\text{mc},i}|$ be the maximum absolute value of the motion control law. Then, for $\|s_i - s_{h_i}\| < r_{\text{com}}$ it follows that

$$\dot{V}_i = \nabla f_i \cdot (u_{\max} - \nabla f_i^{\mathrm{T}}) \,. \tag{2}$$

Since a component of $u_{\max}$ orthogonally to ∇f_i^{T} does not influence $\dot{V}_i$, it is sufficient to consider the case when ∇f_i and $u_{\max}$ are collinear and it follows that

$$\dot{V}_i \leq \|\nabla f_i^{\mathrm{T}}\| \cdot (\|u_{\max}\| - \|\nabla f_i^{\mathrm{T}}\|) < 0 \tag{3}$$

if $\|\nabla f_i^{\mathrm{T}}\| > \|u_{\max}\|$. Due to the properties of f this will be true for $\|s_i - s_{h_i}\| > d'_i$ with $0 < d'_i < r_{\text{com}}$. In other words, the distance between the agent and its downstream neighbor cannot become larger than $d'_i < r_{\text{com}}$ if both are connected once, hence connectivity is maintained.

The proposed control law attracts the agent towards its downstream neighbor if the distance between them becomes too large. Changes of the downstream neighbor result in an actually non-smooth behavior of the control law. However, since the topology controller ensures that a newly picked downstream neighbor is connected to the agent, the connectivity is always guaranteed.

An improved behavior of the connectivity maintenance algorithm will be achieved by the now stated full controller $u_{\mathrm{cc},i}$. Suppose that the downstream neighbor h_i moves away from the stationary agent i. Then, at a certain distance between both agents, agent i will be attracted towards its downstream neighbor and follow its movement. However, if h_i is stationary and agent i moves away, then at a certain distance the connectivity maintenance controller will completely compensate the movement generated by the motion controller and agent i will come to a halt. This limitation in degree of agent i's freedom is overcome by the connectivity maintenance controller stated as follows.

Theorem 2. Let f be a potential function satisfying the above stated properties. Then the connectivity control law

$$u_{\mathrm{cc},i} = -\sum_{j\in\mathcal{N}_i^*} \left.\frac{\partial f}{\partial s_i}\right|_{s_i - s_j}^{\mathrm{T}} \tag{4}$$

maintains the connectivity of agent i with all its critical neighbors $\mathcal{N}_i^*$ under any finite valued motion control law $u_{\mathrm{mc},i}$.

Proof. Due to its similarity to the proof of Lemma 1 and its lengthy character, only a sketch of the proof will be shown. An elaborate proof was shown by Trachte (2008). By using again the Lyapunov function $V_i(s_i) = f(s_i - s_{h_i})$ it can be shown that all critical neighbors can be replaced by a single, virtual critical neighbor and that connectivity with this neighbor is preserved by the connectivity control input $u_{\mathrm{cc},i}$. Because of the smooth and monotonic character of the potential function f, this implies that the connection to all real critical neighbors is maintained.

The connectivity control law $u_{\mathrm{cc},i}$ ensures connectivity with all critical neighbors and thus connectivity of the complete network. Similar to the basic algorithm above, changes in topology introduce non-smoothness to the behavior of the control input. However, the topology controller ensures that newly picked neighbors do not result in undefined values of the connectivity controller. Both controller together result in a minimum restrictive connectivity maintenance algorithm.

Proposition 3. The presented topology control algorithm combined with the connectivity (motion) control law $u_{\mathrm{cc},i}$ results in a minimum restrictive connectivity maintenance algorithm. Minimum restrictive refers here to the fact that the primary movement is constrained by the algorithm as little as possible.

The ability of the algorithm to maintain the connectivity of the network has been shown above. The minimum restrictiveness can be easily illustrated. Suppose a moving agent i with a set of neighbors $\mathcal{N}_i$ and the critical neighbors $\mathcal{N}_i^*$ which, for the sake of simplicity, are all stationary. When the agent reaches a position where the connectivity control law $u_{\mathrm{cc},i}$ increases to a level that might appreciably influence the movement of the agent, three cases are possible:

(i) The increased distance between the agent and some of its critical neighbors leads to a significantly increased virtual communication cost. Therefore, if possible, the shortest path search will select a modified set of critical neighbors such that the value of the connectivity control $u_{\mathrm{cc},i}$ is reduced.

(ii) The agent begins pulling its critical neighbors in the direction of its movement. This is particularly the case, if (i) is not possible and the distances between agent i and its critical neighbors increase towards r_{com}.

(iii) The critical agents cannot move because to motion constraints imposed by their respective critical neighbors. Then the agent will move to a location where the motion control law $u_{\mathrm{mc},i}$ and the connectivity control law $u_{\mathrm{cc},i}$ will completely compensate each other and the agent will also come to a halt.

It becomes clear, that the primary movement of agent i defined by its motion controller $u_{\mathrm{mc},i}$ is only constrained if connectivity of the network cannot be ensured otherwise.

4.3 Extension: maintaining k-connectivity

An interesting extension to the algorithm is to maintain k-connectivity in a network. A graph is k-connected, if there exist at least k disjunct paths between each pair of vertices, i.e. if no two paths share the same edge.

Proposition 4. Consider a multi-agent network as described above that is k-connected at initial time. The k-times parallel execution of the topology control algorithm – particularly of the shortest path search, where the usage of the same downstream neighbor for two or more shortest paths is forbidden – results in k sets $\mathcal{N}_{i,k}^*$ of critical neighbors. Let the union of these sets be denoted as $\mathcal{N}_{i,\cup}^* = \cup_{l=1}^{k} \mathcal{N}_{i,l}^*$, then k-connectivity of the complete network is maintained by computing the connectivity control input $u_{\mathrm{cc},i}$ based on the set of critical neighbors $\mathcal{N}_{i,\cup}^*$.

Maintaining k-connectivity enhances the robustness of the network because it guarantees that $k-1$ links or nodes can fail without breaking the connectivity of the network. Moreover, it directly enhances the algebraic connectivity of the network. Algebraic connectivity is a measurement of the possible information flow in a network. It is important in applications like for example multi-agent consensus problems or distributed decision making, see for instance the work of Wagenpfeil (2008). Since the worst-case lower bound on the information propagation speed depends on the connectivity of the network, maintaining k-connectivity is a suitable method to guarantee a minimal performance of algorithms that rely on distributed information.

5. SIMULATIONS

In this section, the effectiveness of the proposed algorithm should be illustrated by means of a computer simulation. The development over time is shown in figure 3 from left to right and top to down starting at simulation time $t = 0\,\mathrm{s}$ until $t = 7\,\mathrm{s}$ in $1\,\mathrm{s}$ steps. The last image shows the network after $20\,\mathrm{s}$. The network consists of four immobile agents, forming a straight line in the lower left area and four mobile agents. Two of the mobile agents are equipped with a detector (shown in green) that enables them to sense the red depicted gradient field. As those two agents move towards the center of the gradient, the distance to the other two mobile agents converges to $r_{\mathrm{com}} = 5\,\mathrm{m}$, but

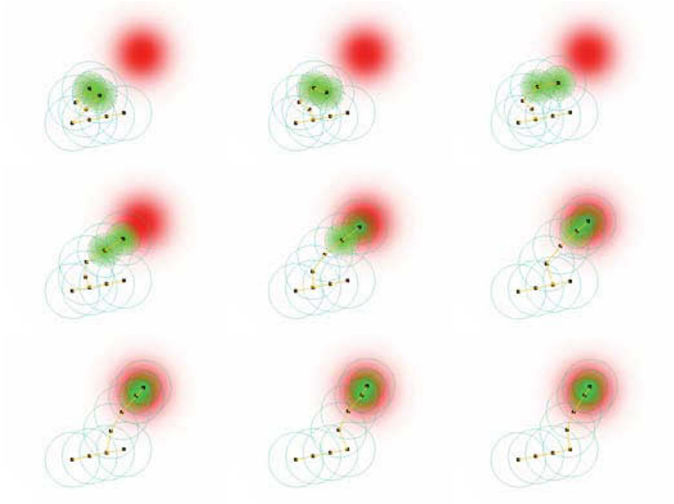

Fig. 3. Simulation of a coverage example.

before the link disconnects, the connectivity maintenance controller lets the agents follow the sensing agents (starting at $t = 3\,\mathrm{s}$). As the mobile agents move along the gradient, the topology controller optimizes the spanning tree and it appears that the lower mobile agent "hops" along the line formed by the stationary agents. When the agents equipped with sensors reach the center of the gradient field, their movement stops and as expected also the agents without sensor stop because the connectivity maintenance control goes to standby. The simulation show the ability of the proposed algorithm to dynamically adjust the topology and position of idle agents to optimize the motion of the network.

6. CONCLUSIONS

In this work, a novel connectivity maintenance algorithm was proposed, that can easily be used to guarantee the connectivity of a mobile multi-agent network. The algorithm can be implemented in a distributed way to execute at each individual agent, since it requires only locally available information. The algorithm combines a discrete topology control algorithm with a continuous connectivity controller. While the topology controller determines which of each agent's neighbors are essential for the connectivity of the network, the location of the agent is constrained by the connectivity controller such that the links to these critical neighbors are preserved. Moreover, the algorithm can easily be extended to maintain k-connectivity. The here proposed connectivity maintenance algorithm differs significantly from previous work by the fact that arbitrary finite-valued motions are covered and that the influence on the spatial behavior of the network is minimal.

A possible next step for future research is the study of networks where the agents' communication devices have individual communication ranges. Although an easy approach is given by using the minimal communication range in an implementation as discussed in this work, it would be of further interest to investigate under which conditions or modifications to the presented algorithm, the full spectrum of communication ranges could be used. Since the shortest path search of the topology controller requires to define one agent as the root node, it would be promising to investigate mechanisms that establish a kind of dynamic root node election. By that, the distance between the root and the outermost agents could be minimized, which in turn could significantly decrease the convergence time of the shortest path search. It would be furthermore interesting to investigate the influence of communication disturbances and an experimental evaluation would be desirable. The proposed algorithm defines a basis for a multitude of further research possibilities, but already defines a well applicable distributed algorithm to maintain connectivity of mobile multi-agent networks.

REFERENCES

Bettstetter, C. (2002). On the minimum node degree and connectivity of a wireless multihop network. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, 80–91. ACM, New York, NY, USA.

Bullo, F., Cortés, J., and Martínez, S. (2008). *Distributed Control of Robotic Networks*. Princeton University Press.

Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. (2002). Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, 8(5), 481–494.

Lynch, N. (1996). *Distributed Algorithms*. Morgan Kaufmann.

Poduri, S. and Sukhatme, G. (2004). Constrained coverage for mobile sensor networks. *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, 1, 165–171 Vol.1.

Spanos, D. and Murray, R. (2004). Robust connectivity of networked vehicles. *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, 3, 2893–2898 Vol.3.

Srivastava, K. and Spong, M. (2008). Multi-agent coordination under connectivity constraints. In *Proc. American Control Conference*, 2648–2653.

Trachte, A. (2008). *Dynamic Shortest Path Search and Connectivity Maintenance in Multi-Agent Systems*. Master's thesis, University Stuttgart. URL `http://fujita.fl.ctrl.titech.ac.jp/researches/movie/diploma_theses.zip`.

Wagenpfeil, J. (2008). *Consensus Problems and Decision Making in Multi-Agent Networks with Range-Limited Information Exchange*. Master's thesis, University Stuttgart.

Wang, X., Xing, G., Zhang, Y., Lu, C., Pless, R., and Gill, C. (2003). Integrated coverage and connectivity configuration in wireless sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, 28–39. ACM, New York, NY, USA.

Xue, F. and Kumar, P.R. (2004). The number of neighbors needed for connectivity of wireless networks. *Wireless Networks*, 10, 169–181.

Zavlanos, M. and Pappas, G. (2005). Controlling connectivity of dynamic graphs. *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, 6388–6393.

Zavlanos, M. and Pappas, G. (2007). Distributed connectivity control of mobile networks. In *Proc. 46th IEEE Conference on Decision and Control*, 3591–3596.