2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

Distributed decision making for task switching via a consensus-like algorithm

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

***Abstract*—We propose a consensus-like algorithm for switching from a first to a second task in Multi-Agent Systems. The task-switch is supposed to occur synchronously for all agents after each single agent has finished the first task, because the two tasks may influence each other. As every agent has only local information about its own task-state, the proposed algorithm is used to determine the state of the whole network. With this additional information each agent can decide when to safely switch the task. In this paper, the first task is a distributed exploration control law, while the second task is a distributed coverage control law where the goal is to maximize the probability of detecting certain events. By exploring the mission space before switching to the coverage task, the overall probability of detecting those events is increased. For the exploration task a r-limited voronoi cell based algorithm is used and for the coverage task a gradient based control law.**

***Index Terms*—multi-agent networks, distributed algorithms, coverage control, exploration, synchronous taskswitch, decision making**

I. INTRODUCTION

Multi-Agent Systems are systems in which several interacting intelligent agents pursue some set of goals or perform some set of tasks. These could be motion coordination tasks as proposed in [1], like rendezvous [2] [3], deployment [4], coverage [4] [5] or flocking control [2] [6]. For some goals it is of interest to switch between several tasks due to changing conditions or a goal which can be reached more easily by switching from one task to another. This could be for example a group of agents, which have to move to a certain position, using flocking control for the movement, and switch then to a coverage or exploration task when they reached their final destination. Another goal would be to first explore the area, to gather relevant information about the environment and then switch to a coverage task, to cover regions of high interest.

We will focus upon a taskswitch from exploration to coverage in this paper, nevertheless the algorithm can be used also for any other taskswitch matching the later stated assumptions. The taskswitching problem is motivated by the occurrence of local maxima in the coverage control problem defined by Li and Cassandras [7]. They introduced a sensor network of agents, equipped with communication devices and a single base station as a central data processing unit with the goal to maximize the detected events that take place randomly in the mission space. This coverage control problem is formulated as a cost function that describes the expected event detection probability over the whole mission space. To maximize the joint event detection probability, Li and Cassandras developed a gradient based algorithm which requires only local information that every agent is able to gather. As the distributed control algorithm uses local gradients to compute the movement of the agents, it is only possible to cover local maxima while other regions with high event frequency may not be covered. To increase the coverage and the event detection probability over the whole mission space, it is reasonable to explore the mission space in a first task and then switch the task to coverage as the second task. Thereby, all regions with a high density function will be discovered and, after switching the task, covered by the sensing agents by using the gradient based algorithm.

In this paper we will introduce a consensus-like algorithm and exemplify its usage by realizing a taskswitch from exploration to coverage control. The taskswitch is supposed to occur when the first task is finished, which is generally a global condition, as the switch from exploration to coverage should happen after all agents finished exploring the mission space. But as agents can gather information only locally, by sensing or communicating with their neighbors, it is necessary to achieve a taskswitch consenus of all agents by using only the provided local information. For the exploration task a r-limited voronoi partition algorithm is used as proposed in [1] and [6], where all agents move to the centroid of their voronoi partition, and for the coverage task the gradient based control law proposed by Li and Cassandras [7] is used.

II. MOTIVATION AND PROBLEM SETUP

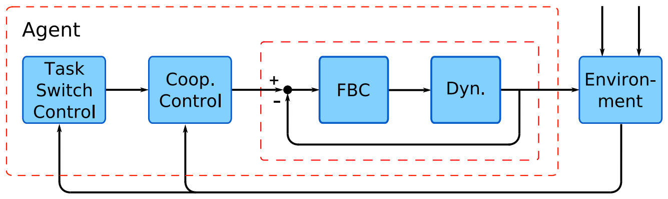

Fig. 1: Model of the agent

The mission space is a simply connected area $\Omega \subset \mathbb{R}^2$, wherein a network of n agents with the distinct identifiers $\mathcal{I} = \{1, 2, \ldots, n\}$ exists. The position of each agent i is given by $s_i \in \Omega$ and all agents positions are described by the vector $s = [s_1, s_2, \ldots, s_n]$. Agent i's dynamics is defined

Jay Wagenpfeil, Adrian Trachte and Oliver Sawodny are with the Institute for System Dynamics, University of Stuttgart, 70569 Stuttgart, GERMANY, jay.wagenpfeil@isys.uni-stuttgart.de

Takeshi Hatanaka and Masayuki Fujita are with the Department of Mechanical and Control Engineering, Tokyo Institute of Technology, Tokyo 152-8552, JAPAN, hatanaka@ctrl.titech.ac.jp

by the simple differential equation $\dot{s}_i = u_i$, with u_i being the input.[1] One of the agents is fixed and denoted as the base station. A feedback-controller (FBC) makes the agent follow reference trajectories that are generated by a task-dependent cooperative control law. The task-state variable z_i indicates if agent i is still working on the first task ($z_i = 0$) or has already finished it ($z_i = 1$). The consensus variable x_i is used to determine the task-state of the whole network. The task switch is controlled by a dedicated module that is located in the outermost layer of the agent model, which can be seen in figure 1.

A. Exploration Task

For the exploration task an algorithm based on r-limited Voronoi partitions is used as proposed in [4]. The r-limited Voronoi partition of an agent i is defined by $V_i = \{q \in \Omega \mid \|q - s_i\| \leq \min(\|q - s_j\|, R_{\text{vc}})\ \forall\, j \in \mathcal{I}\}$, where R_{vc} is the radius. The derivative of the *multi-center function* $\mathcal{H}_{\text{C}}(s)$ defined in [4] with respect to each agents position s_i can be computed, with $f(q) = -q^2$ as performance function, as:

$$\begin{aligned}\frac{\partial \mathcal{H}_{\text{C}}}{\partial s_i}(s) &= 2\int_{V_i} (q - s_i)\phi(q)dq \\ &= 2m_{V_i}(cm_{V_i} - s_i)\,,\end{aligned} \tag{1}$$

where $\phi(q)$ is the in [4] defined density function, and m_{V_i} and cm_{V_i} are the mass and the center of mass of the Voronoi cell with respect to the density function $\phi(q)$. But as we are in our exploration task only interested in a maximum deployment of the agents, we set the density function $\phi(q) = 1\ \forall q \in \Omega$. Therefore the terms m_{V_i} and cm_{V_i} equal now the area of the Voronoi partition and the center of area. With this we can define the input for the exploration task as:

$$u_{i,e} = \frac{\partial \mathcal{H}_{\text{C}}}{\partial s_i} = 2m_{V_i}(cm_{V_i} - s_i) + u_{i,\text{ktf}} \tag{2}$$

$u_{i,\text{ktf}}$ is a term to guarantee connectivity (see the Appendix). The convergence of agents with the simple dynamics $\dot{s}_i = u_i$, and the exploration task input $u_{i,e}$ with $u_{i,\text{ktf}} = 0$ is proven in [4]. We assume that the agents can deploy over the whole area Ω without losing connectivity and that the term $u_{i,\text{ktf}}$ does therefore not interfere with the convergence of the exploration task.[2] To determine if the exploration has been finished, the threshold ϵ_{exp} is defined and each agent i sets its task-state variable z_i according to

$$z_i = \begin{cases} 0 & \text{if } \|u_{i,e}\| > \epsilon_{\text{exp}} \\ 1 & \text{if } \|u_{i,e}\| \leq \epsilon_{\text{exp}}\,. \end{cases} \tag{3}$$

[1]In many real-world setups non-holonomic motion constraints appear. For the sake of generality, in this paper we consider only a simple motion dynamics. A simple approach is to introduce an additional controller between the feedback-controller and the cooperative-control module, which locally compensates the non-holonomically constrained system such that the reference trajectory for a single integrator model can be applied.

[2]In the case that the mission space cannot be completely explored without losing connectivity, a steady state formation of the agents is reached, which is the maximum deployment within the connectivity constraints.

B. Coverage Task

For the coverage task, the agent will follow a gradient, computed to maximize the cost function $\mathcal{H}_{\text{CS}}(s) = \int_\Omega E(x)P(x,s)dx$ [3] introduced by Li and Cassandras [7]. The cost function describes the joint-event detection probability with $E(x)$ defining the event density function, which is the frequency with which random events take place (Hz/m^2). $P(x,s)$ is the probability that an event is detected by the sensor network and is defined as

$$P(q,s) = 1 - \prod_{i=1}^{n}[1 - p_i(q)]\,, \tag{4}$$

where $p_i(q)$ is the probability of each agent to detect an event. With the assumption, that the detection probability to detect events of an agent $p_i(q)$ is equal to zero if the range $\delta_i(q) = \|q - s_i\|$ is bigger than the maximum sensing range R_{sens}, it is possible to define $\Omega_i = \{q \mid \delta_i(q) \leq R_{\text{sens}}\}$, which is the sensing area of agent i, and maximizes the cost function by using the following local gradient:

$$\frac{\partial \mathcal{H}_{\text{CS}}}{\partial s_i} = \int_{\Omega_i} \Phi(q) \prod_{k \in \mathcal{N}_i \setminus \{i\}} [1 - p_k(q)] \frac{dp_i(q)}{d\delta_i(q)} \frac{s_i - q}{\delta_i(q)} dq \tag{5}$$

Note that this means that each agent i must be able to locate its neighbor agents at least within the distance $2R_{\text{sens}}$. Locating the agents is independent from sensing the environment and thus not limited to the sensing range of the detector. With (5) the input for the coverage task can be defined as:

$$u_{i,c} = \omega_1 \frac{\partial \mathcal{H}_{\text{CS}}}{\partial s_i} - \omega_2 \frac{\partial \mathcal{H}_{\text{COM}}}{\partial s_i} + u_{i,\text{ktf}} \tag{6}$$

where ω_1 and ω_2 are weighting factors. G is an additional communication cost term, which can be found in [7] and $u_{i,\text{ktf}}$ is a term to guarantee connectivity (see the Appendix).

C. Communication

Each agent is equipped with a communication device that enables it to establish a bi-directional communication with its neighbors $\mathcal{N}_i = \{j \in \mathcal{I} : \|s_i - s_j\| < R_{\text{com}}\}$ that are the agents within communication range R_{com}. Note that this implies that each agent has itself as a neighbor. To evaluate the task-state of the network, agents communicate with all neighboring agents during the exploration task. In discrete time, this communication topology can be modelled at each time-step k by the undirected unweighted symmetric graph $\mathcal{G}(k) = (\mathcal{V}, \mathcal{E}(k))$, with the vertices $\mathcal{V} = \mathcal{I}$ and the possibly time-varying set of edges $\mathcal{E}(k) \subseteq \mathcal{I} \times \mathcal{I}$ which also includes self-loops. The pair (i,j) is an edge of the graph if and only if agent i is a neighbor of agent j (and vice versa). In every time-step we can define the adjacency matrix $A(k) = [a_{ij}(k)]$ to the graph $\mathcal{G}(k)$, which is a symmetric $n \times n$ matrix. For the elements $a_{ij}(k)$ holds that $a_{ij}(k) = 1$ if $(i,j) \in \mathcal{E}(k)$, and $a_{ij}(k) = 0$ otherwise. Additionally we define $d_i(k) = \|\mathcal{N}_i\|$, which is the number of neighbors of each agent in each time

[3]Note that in [7], Li and Cassandras use R(x) instead of E(x) for the event density function.

step, and the matrix $D(k)$, which is a diagonal matrix with $d_i(k)$ as entries.

There is a secondary, overlying communication protocol that computes the cheapest path (with respect to the communication cost) from each agent to the base station. The thereby formed communication graph has a tree structure and can thus not only be used to minimize the communication costs during the coverage task but also to ensure connectivity of the network during both tasks. A detailed study of shortest path searches in dynamically changing networks and the design of a connectivity maintenance control law can be found in diploma thesis of Trachte [8].

Please note that we do not take physical properties of the communication medium into account. Particularly the implementation of wireless communication protocols requires the consideration of issues like interference and other radio disturbances.

III. Algorithm

In this section we will present the algorithm that will control and synchronize the taskswitch of each agent in the network.

Assumption 1: There exists a time k_0 such that $A(k) = A(k_0)$ for all $k \geq k_0$. That means there exists a time after which the communication topology is fixed. This assumption can be motivated by the given problem setup: the task should be switched after all agents have finished the exploration task, i.e. when all agents stopped their movement. Neglecting any disturbance effects on communication, including failure of single agents, it is possible to assume that the communication topology remains fixed, if the agents do not move.

If a change in topology occurs between the time-step k and the following time-step $k+1$, then $A(k+1) \neq A(k)$ or equivalently, the set $\mathcal{I}_{\text{tc}}(k) = \{j : \mathcal{N}_j(k+1) \neq \mathcal{N}_j(k)\}$ is non-empty. For those agents $i \in \mathcal{I}_{\text{tc}}(k)$ that have different neighbors at time-steps k and $k+1$, we set in step k the task-state $z_i(k) = 0$.

Assumption 2: There exists a time $k_1 \geq k_0$ such that $z(k) = \underline{1}$, respectively $Z(k) = I$, for all $k \geq k_1$. This means there exists a time after which all agents in the network have completely finished the first task.

It is furthermore assumed, that the number of agents in the network and the communication range are chosen such that the agents can deploy over the whole mission space. Moreover it is required that connectivity is maintained at all times, this problem is left to the implementation of the control tasks. With these assumptions we can state the algorithm for the synchronized task-switch. The algorithm is derived from a first-order linear consensus protocol and is for each agent i given by

$$x_i(k+1) = z_i(k) \cdot \frac{1}{d_i(k)+1} \cdot \left[\sum_{j \in \mathcal{N}_i(k)} x_j(k) + 1 \right]. \quad (7)$$

The value of the consensus variable x_i of each agent i is in the next step set to the average of its task-state variable and all its neighbors consensus variables if the agent has finished the first task, or to 0 otherwise. At the end of each step, agents send a message to all their neighbors telling them thereby the state of their consensus variable; these messages are received and processed in the beginning of the next step. While this implies some kind of synchrony between the agents, this is not subject of this paper and might be a task for a underlying layer of the communication protocol. A formal description of the complete sequence of operations within one step is shown in figure 2.

```
begin step
    receive x(k) from neighbors;
    {compute movement; move;}
    set task-state $z_i(k)$;
    if topology changed then
        reset $z_i(k)$;
    end if
    compute $x_i(k+1)$;
    send $x_i(k+1)$ to neighbors;
end step
```

Fig. 2: Sequence of operations within one step in a pseudo code.

For the complete multi-agent network, the update rule for the next step is given by

$$x(k+1) = Z(k) \cdot (D(k)+I)^{-1} \cdot [A(k)x(k) + \underline{1}]. \quad (8)$$

Each agent will use its consensus variable to determine the task-state of the whole network. If not all agents have finished the first task or there are changes in topology, i.e. if $z \neq \underline{1}$, then $x_i(k) < 1$. If all agents finished the first task and the communication topology remains fixed, then the value of the consensus variable of each agent will converge to 1.

A. Convergence of the algorithm

The following lemma will state the convergence of the algorithm. An elaborate proof can be found in the diploma thesis by Wagenpfeil [9]. Related studies of other linear consensus algorithms can be found in the relevant literature.

Lemma 1: Under the stated assumptions, the algorithm will converge such that for all agents $i \in \mathcal{I}$ the consensus variable $x_i(k) \to 1$ for $k \to \infty, k \geq k_1$.

B. Threshold design for switching the task

The presented algorithm will let the consensus variable of each agent converge to 1, when the exploration task is finished. As the convergence is asymptotical, the actual switch from the first to the second task will be performed when the value of the consensus variable is sufficiently close to one. To avoid premature task-switches, a threshold for switching the task must be defined, that is larger than all consensus values that occur at any time while the first task has not yet been finished.

Lemma 2: An upper bound on the value of the consensus variables at any time in a network of n agents is given by the maximal steady-state value $\|\bar{x}^*\|_\infty$ in a worst-case scenario with fixed chain topology, where all agents but one at the end of the chain have finished the exploration task.

For an elaborate proof of this lemma please refer to [9]. By constructing such a network, it is possible to compute the worst case steady-state consensus values for a static network. At steady state, the system is described by

$$\bar{x}^* = \bar{Z}^* \cdot (\bar{D}^* + I)^{-1} \cdot (\bar{A}^* \bar{x}^* + \underline{1}) \,, \tag{9}$$

where $\bar{Z}^* = diag([1\,1\,1 \cdots 1\,0]^T)$ and $\bar{D}^* = diag([3\,4\,4 \cdots 4\,3]^T)$ are diagonal matrices, and $\bar{A}^*$ is a tridiagonal matrix with ones on the main and the two secondary diagonals. Since it is already known that $\bar{x}^*_n = 0$, the last equation – respectively the last row and column of each matrix and the last element of each vector – can be omitted and it follows

$$(\bar{D}_{[1\,\dots\,n-1]} + I_{[1\,\dots\,n-1]} - \bar{A}^*_{[1\,\dots\,n-1]})\bar{x}^*_{[1\,\dots\,n-1]} = \underline{1}_{[1\,\dots\,n-1]} \,.$$

Due to the properties of the matrices A and D, this equation can be solved and the threshold for switching the task is given by $\delta = \|x^*\|_\infty = \bar{x}^*_1$.

Theorem 1: Under Assumption 1, the following statements are true. If $x_i(k) > \delta$ for some i, then $z_j(k) = 1 \;\forall\; j \in \mathcal{I}$, in other words the first task is completed. If $z_j(k) \neq 1$ for some $j \in \mathcal{I}$, then $x_i(k) \leq \delta \;\forall\; i \in \mathcal{I}$.

The proof of this Theorem follows straightly from Lemma 2.

The analysis so far assumed a more or less constant input – i.e. the values of the task-state variables – before the task-switch occurs. The analysis of an arbitrary dynamically changing input is more challenging. For networks with more than two agents the authors could not find an input that destabilizes the system such that the consensus variables take values larger than the proposed maximum steady-state value for a constant input. Unfortunately, a proof that such an input does not exist could not be found. Moreover the case of malicious or faulty agents that generate arbitrary input values should be considered, algorithms to detect and remove such agents are discussed in [10].

IV. Time complexity

Lynch [11] defines the notion of time complexity, which is basically the time that an algorithm needs to perform. In our setup where we want to switch to the coverage task after finishing the exploration task, a sensible definition of the time complexity could be as follows: the time complexity TC is the time from when the last agent finished the exploration task until the last agent starts with the coverage task.

Agents that start with the coverage task may interfere with neighboring agents, that have finished the exploration task but not yet switched to the coverage task. The movement might cause those neighbors to resume the exploration task. To avoid this, agents that switch the task to coverage have to wait a certain time Δk_{safety} to ensure that all agents will have switched to the coverage task during this time. Setting $\Delta k_{\text{safety}} = TC_{\text{wc}}(n)$ to the maximum value of the time complexity $TC_{\text{wc}}(n)$ will guarantee, that no agent starts performing the coverage task before the last agent in the network has switched to the coverage task.

Theorem 2: An upper bound on the time-complexity $TC_{\text{wc}}(n)$ is given by

$$TC_{\text{wc}}(n) = \frac{\ln(1-\delta(n))}{\ln\left(\frac{n}{n+1}\right)} \,, \tag{10}$$

where $\delta(n)$ is the threshold of the task-switch as designed in section III-B depending on the number of agents n.

Proof: Remember that k_1 is the time-step when the last agent finishes the exploration task, i.e. $z(k_1 - 1) \neq \underline{1}$ and $z(k) = \underline{1}$ for all $k \geq k_1$. Let $x_{\text{lb}}(k_1)$ be the lowest value of all consensus variables in the network, i.e. $x_{\text{lb}}(k_1) \leq x_j(k_1)$ for all $j \in \mathcal{I}$. In the next step $k_1 + 1$, for any agent i holds

$$\begin{aligned} x_i(k_1+1) &= \frac{1}{d_i+1} \cdot \left(\sum_{j \in \mathcal{N}_i(k_1)} x_j(k_1) + 1 \right) \\ &\geq \frac{1}{d_i+1} \cdot (d_i \cdot x_{\text{lb}}(k_1) + 1) \\ &\geq \frac{1}{d_{\max}+1} \cdot (d_{\max} \cdot x_{\text{lb}}(k_1) + 1) \end{aligned}$$

where $d_{\max} = \max_{i \in \mathcal{I}}\{d_i\}$, keeping in mind that $x_i < 1$ for all $i \in \mathcal{I}$. A single agent however has no knowledge of the communication topology of the whole network and hence cannot determine $d_{\max}$. Therefore, a worst case topology has to be assumed, which corresponds to the existence of an agent with the largest possible number of neighbors which is $d_{\max,\text{wc}} = n$. The lower bound $x_{\text{lb}}(k_1 + 1)$ to all consensus variables in step $k_1 + 1$ is thus given by

$$x_{\text{lb}}(k_1+1) = \frac{1}{n+1} \cdot (n \cdot x_{\text{lb}}(k_1) + 1) \,.$$

The same argumentation holds for the lower bounds on the consensus variables in the following steps $k_1 + 2$, $k_1 + 3$, *etc.* The lower bound can therefore generally be described by

$$x_{\text{lb}}(k+1) = \frac{1}{n+1} \cdot (n \cdot x_{\text{lb}}(k) + 1) \tag{11}$$

with $k \geq k_1$ and $x_{\text{lb},0} := x_{\text{lb}}(k_1) = \min_i\{x_i(k_1)\}$. This is a simple difference equation with the solution

$$x_{\text{lb}}(k) = 1 - \left(\frac{n}{n+1}\right)^{k-k_1} (1 - x_{\text{lb},0}). \tag{12}$$

With (12) it is possible to compute the latest possible time $t_{\text{cov,wc}}$ at which all agents have started the coverage task[4] . Using the above defined threshold $\bar{x}^*_1$, then from the switching condition for the lower bound on the consensus variables follows

$$t_{\text{cov,wc}} = \inf_{t \in \mathbb{R}} \left\{ 1 - \left(\frac{n}{n+1}\right)^{t-k_1} (1 - x_{\text{lb},0}) \geq \bar{x}^*_1 \right\}.$$

[4]Note that $t_{\text{cov,wc}} \in \mathbb{R}$ for the sake of an easier notation. For the discrete-time setup, the worst-case time-step $k_{\text{cov,wc}}$ is given by $k_{\text{cov,wc}} = \inf\{k \in \mathbb{Z} \,|\, k \geq t_{\text{cov,wc}}\}$.

center μ	width σ	peak value α
(18,10)	6	0.5
(25,20)	4	1.0
(55,05)	6	1.0
(45,15)	4	0.5
(60,28)	6	1.0

TABLE I: Parameters for the event density function, center position μ and width σ in meters.

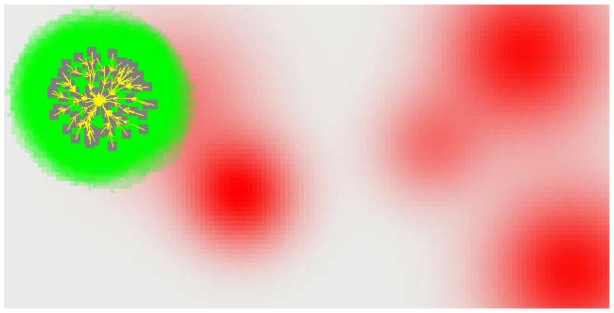

Fig. 3: Deployment of 100 agents into the mission-space at time $t = 0.1\,\mathrm{s}$

The worst initial condition for the lowest bound clearly is $x_{\mathrm{lb},0} = 0$ and it follows

$$t_{\mathrm{cov,wc}} = \frac{\ln(1-\delta(n))}{\ln\left(\frac{n}{n+1}\right)} + k_1, \tag{13}$$

respectively for the upper bound on the time complexity, which is given by $TC_{\mathrm{wc}} = t_{\mathrm{cov,wc}} - k_1$, follows (10). ■

V. Simulation results

The previously presented problem setup has been implemented in a Java-based multi-agent simulation environment to illustrate the benefit of the proposed algorithm. The mission space is a rectangular shaped area with the dimensions $64\,\mathrm{m} \times 32\,\mathrm{m}$. The event density function $E(x)$ is given by the sum of five rotationally symmetric 2-dimensional Gaussian functions of the form $f_{\mathrm{gauss}}(x) = \alpha \cdot exp\left(-\frac{\|x-\mu\|^2}{2\sigma^2}\right)$ (in $\mathrm{Hz/m^2}$). The parameters for each of the five Gaussian functions can be found in table I. The simulation is run with a fixed sample time of 0.02s with time-discrete versions of the agents dynamics and control laws, derived from the equations in section II.

A network of 100 mobile agents and one base is deployed within a disc with radius 5 m located at $(10\,\mathrm{m}, 10\,\mathrm{m})$. Figure 3 shows the agents at time $0.1\,\mathrm{s}$, the agents are depicted as blue squares and intensity of the red color corresponds to the event density function. Each mobile agent is equipped with a sensor to detect nearby occurring events. The sensing range of this detector is $R_{\mathrm{sens}} = 5\,\mathrm{m}$ and the detection probability, depicted in green, is modeled by $p_i(x) = e^{-\|x-s_i\|}$ for all agents $i \in \mathcal{I}$. The communication range is limited to $R_{\mathrm{comm}} = 21\,\mathrm{m}$.

We will present simulation results first for a classical setup like in [7], where the agents only perform the coverage task, and second for the extended setup where the agents will switch to the coverage task after completely exploring the mission space.

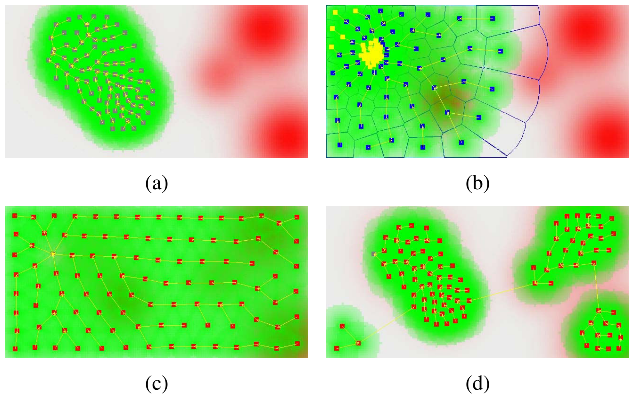

Fig. 4: **(a)** Agents at stationary formation (shown at time $t = 60\,\mathrm{s}$) when only using the coverage algorithm by Li and Cassandras [7]. **(b)-(d)** When using the proposed algorithm, agents first explore the mission space (**(b)** at $t = 4\,\mathrm{s}$), synchronously switch the task (**(c)** at $t = 43\,\mathrm{s}$) and converge to a stationary formation (**(d)** at $t = 60\,\mathrm{s}$)

.

In the first case, the network finally converges to a stationary formation as shown in figure 4(a). It can be clearly seen that several areas of high event density are not covered.

In the second case, the agents first deploy over the whole mission space, as seen in figure 4(b), using r-limited Voronoi cells (borders are depicted in blue) with radius $R_{\mathrm{vc}} = 10\,\mathrm{m}$. Agents that stop or come close to a rest are considered to have finished the first task and are depicted as yellow squares. After all agents have finished the first task, they synchronously switch the task according to the presented algorithm (as seen in figure 4(c), agents are now depicted as red squares) and start with the coverage task. The network converges to a stationary formation that clearly differs from the one seen in the first case. Figure 4(d) shows that now also the areas with high event density in the right half plane of the mission space are well covered by agents. There is a small group of three agents in the lower left corner of the mission space, that remains stationary at this position. The observed behavior occurs because the gradient of the event density function is very close to zero and so are the generated reference trajectories for these agents. This is another drawback of the original gradient-based coverage algorithm, but is not to be addressed in this work.

Figure 5 illustrates the joint rate of events detected by all agents. In the first case for only coverage the joint event detection rate reaches a maximum value of $591.0\,\mathrm{Hz}$. For the extended setup with exploration the joint event detection rate converges to a value of $596.7\,\mathrm{Hz}$ during the exploration task. After switching to the coverage task at time $t = 41.84\,\mathrm{s}$, $F(s)$ further increases and reaches a top value of $994.1\,\mathrm{Hz}$. Interestingly, before the task-switch happens, both setups have almost the same event detection rate. This means, that the exploration task reaches the same value as the gradient based coverage task. After switching from exploration to

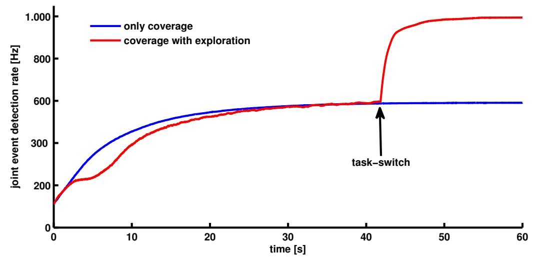

Fig. 5: The joint event detection rate for a setup with only coverage (blue) and an extended setup with preceded exploration (red).

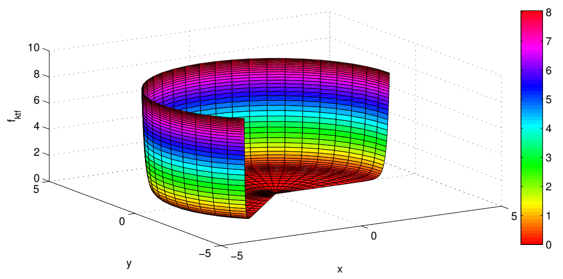

Fig. 6: An exemplary potential function $f_{\text{ktf}} = \|\eta\| \cdot (R_{\text{com}} - \|\eta\|)^{-10}$ with $R_{\text{com}} = 5$, $d = 2$, $\eta = \|[x, y]^T\|$ and function values above 8 are cut off.

coverage the second setup reaches a value which is about 68% higher than the setup using only coverage.

VI. Conclusions

In this paper, an algorithm to control the synchronous task-switch of all agents in a mobile network is proposed. The decision when to switch the task of an individual agent is made solely based on locally available information. Therefore, the algorithm can easily be implemented and does not require broadcasting mechanisms to communicate with all agents in a network. Instead, the necessary information exchange within the network relies only on communication with direct neighbors. The convergence of the algorithm was closely investigated and an upper bound of the time complexity was shown. The benefit of the proposed algorithm was shown by combining a cooperative sensing problem with an exploration task. By that it was possible to significantly increase the stationary joint event detection rate in a java-based simulation.

Appendix

Connectivity maintenance is based on the cheapest path communication protocol, which uses a shortest path algorithm (like e.g. in [12]) to find the cheapest communication path from each agent through the network to the base. As long as such a path can be found for each agent, the whole network remains connected. Each agent i (except for the base station) has a downstream neighbor h_i, that is the next hop on the path towards the base. Agents whose downstream neighbor is agent i, are agent i's upstream neighbors and denoted as $\mathcal{U}_i$.

Connectivity to the base is maintained as long as all agents stay connected with their down- and upstream neighbors. This is achieved by means of a so called keep-together function $f_{\text{ktf},i}(R_{\text{com}}, \eta_i)$, with $\eta_i = \|s_{h_i} - s_i\|$ being the distance between agent i and its downstream neighbor agent h_i. This artificial potential function is designed such that it is close to zero for $\eta_i < R_{\text{com}}$ and $f_{\text{ktf},i} \to \infty$ for $\eta_i \to R_{\text{com}}$ (e.g as used for the simulations $\|\eta\| \cdot (R_{\text{com}} - \|\eta\|)^{-10}$). By adding the term

$$u_{i,\text{ktf}} = -\left(\frac{\partial f_{\text{ktf},i}}{\partial s_i}\right)^{\text{T}} - \sum_{j\in\mathcal{U}_i}\left(\frac{\partial f_{\text{ktf},j}}{\partial s_i}\right)^{\text{T}} \tag{14}$$

to the input u_i of each agent, connectivity to the down- and upstream neighbors is achieved.

The keep-together-function can be thought of as a chain link between the agents. As long as each agent is close enough to all of its neighbors the influence of the keep-together-function is low, as the function is close to zero. But as soon as an agent comes close to its communication range and is endangered to lose the connectivity to one of its neighbors, the keep-together function pulls the agent towards this neighbor and vice-versa.

References

[1] S. Martinez, J. Cortes, and F. Bullo, "Motion coordination with distributed information," *Control Systems Magazine, IEEE*, vol. 27, no. 4, pp. 75–88, 2007.
[2] J. Lin, A. Morse, and B. Anderson, "The multi-agent rendezvous problem," in *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, vol. 2, 2003, pp. 1508–1513 Vol.2.
[3] J. Cortes, S. Martinez, and F. Bullo, "Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions," *Automatic Control, IEEE Transactions on*, vol. 51, no. 8, pp. 1289–1298, 2006.
[4] ——, "Spatially-distributed coverage optimization and control with limited-range interactions," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 4, pp. 691–719, 2005.
[5] M. Schwager, J.-J. Slotine, and D. Rus, "Decentralized, adaptive control for coverage with networked robots," in *Robotics and Automation, 2007 IEEE International Conference on*, 10-14 April 2007, pp. 3289–3294.
[6] Q. Jiang, "An improved algorithm for coordination control of multi-agent system based on r-limited voronoi partitions," *Automation Science and Engineering, 2006. CASE '06. IEEE International Conference on*, pp. 667–671, Oct. 2006.
[7] W. Li and C. Cassandras, "Distributed cooperative coverage control of sensor networks," in *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, 12-15 Dec. 2005, pp. 2542–2547.
[8] A. Trachte, "Dynamic shortest path search and connectivity maintenance in multi-agent systems," Master's thesis, University Stuttgart, 2008.
[9] J. Wagenpfeil, "Consensus problems and decision making in multi-agent networks with range-limited information exchange," Master's thesis, University Stuttgart, 2008.
[10] M. Jelasity, A. Montresor, and O. Babaoglu, "Detection and removal of malicious peers in gossip-based protocols," in *In Future Directions in Distributed Computing 2004*, 2004.
[11] N. Lynch, *Distributed Algorithms*. Morgan Kaufmann, 1996.
[12] J. Al-Karaki and A. Kamal, "Routing techniques in wireless sensor networks: a survey," *Wireless Communications, IEEE [see also IEEE Personal Communications]*, vol. 11, no. 6, pp. 6–28, 2004.