Self-triggered coordination of robotic networks for optimal deployment

C. Nowzari [a] J. Cortés [a]

[a] *Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA, 92093, USA*

Abstract

This paper studies a deployment problem for a group of robots where individual agents operate with outdated information about each other's locations. Our objective is to understand to what extent outdated information is still useful and at which point it becomes essential to obtain new, up-to-date information. We propose a self-triggered coordination algorithm based on spatial partitioning techniques with uncertain information. We analyze its correctness in synchronous and asynchronous scenarios, and establish the same convergence guarantees that a synchronous algorithm with perfect information at all times would achieve. The technical approach combines computational geometry, set-valued stability analysis, and event-based systems.

Key words: robotic networks, self-triggered control, spatial partitioning, uncertain information, set-valued analysis

1 Introduction

This paper studies a robotic sensor network performing an optimal static deployment task when individual agents do not have up-to-date information about each other's locations. Our objective is to design a self-triggered coordination algorithm where agents autonomously decide when they need new, up-to-date location information in order to successfully perform the required task. Our motivation comes from the need for strategies that naturally account for uncertainty in the state of other agents and are able to produce substantial energy savings in the operation of the network.

Literature review. There are two main areas related to the contents of this paper. In the context of robotic sensor networks, this work builds on [Cortés et al., 2004], where distributed algorithms based on centroidal Voronoi partitions are presented, and [Cortés et al., 2005], where limited-range interactions are considered. Other works on deployment coverage problems include [Howard et al., 2002, Schwager et al., 2009, Kwok and Martínez, 2010, Pavone et al., 2011]. We note that the locational optimization problem considered here is a *static* coverage problem, in contrast to *dynamic* coverage problems, e.g., [Choset, 2001, Hussein and Stipanovič, 2007], that seek to visit or continuously sense all points in the environment. A feature of the algorithms mentioned above is the common assumption of constant communication among agents and up-to-date information about each other's locations.

The other area of relevance to this work is discrete-event systems [Cassandras and Lafortune, 2007], and the research in triggered control [Velasco et al., 2003, Subramanian and Fekri, 2006, Wang and Lemmon, 2009, Anta and Tabuada, 2010], particularly as related to sensor and actuator networks. Of particular relevance are works that study self-triggered or event-triggered decentralized strategies that are based on local interactions with neighbors defined in an appropriate graph. Among them, we highlight [Kang et al., 2008] on collision avoidance while performing point-to-point reconfiguration, [Dimarogonas and Johansson, 2009] on achieving agreement, [Wan and Lemmon, 2009] on distributed optimization, and [Mazo Jr. and Tabuada, 2011] on implementing nonlinear controllers over sensor and actuator networks. This paper shares with these works the aim of trading computation and decision making at the agent level for less communication, sensing or actuator effort while still guaranteeing a desired level of performance.

Statement of contributions. The main contribution of the paper is the design of the `self-triggered centroid algorithm` to achieve optimal static deployment in a given convex environment. This strategy is based on two building blocks. The first building block is an update policy that helps an agent determine if the information it possesses about the other agents is sufficiently up-to-date. This update policy is based on spatial partitioning techniques with uncertain informa-

Email addresses: `cnowzari@ucsd.edu` (C. Nowzari), `cortes@ucsd.edu` (J. Cortés).

tion, and in particular, on the notions of guaranteed and dual guaranteed Voronoi diagrams. The second building block is a motion control law that, given the (possibly outdated) information an agent has, determines a motion plan that is guaranteed to contribute positively to achieving the deployment task. To execute the proposed algorithm, individual agents only need to have location information about a (precisely characterized) subset of the network and in particular, do not need to know the total number of agents. We establish the monotonic evolution of the aggregate objective function encoding the notion of deployment and characterize the convergence properties of the algorithm. Due to the discontinuous nature of the data structure that agents maintain in our self-triggered law, the technical approach resorts to a combination of tools from computational geometry, set-valued analysis, and stability theory. We show that both synchronous and asynchronous executions of the `self-triggered centroid algorithm` asymptotically achieve the same optimal configurations that an algorithm with perfect location information would, and illustrate their performance and cost in simulations.

Organization. Section 2 outlines some important notions from computational geometry. Section 3 contains the problem statement. Section 4 introduces the notions of guaranteed and dual guaranteed Voronoi diagrams. Section 5 presents our algorithm design and Section 6 analyzes the convergence properties of its synchronous executions. Section 7 discusses an extension to further reduce communication costs and the convergence of asynchronous executions. Simulations illustrate our results in Section 8. We gather our conclusions in Section 9.

2 Preliminaries

Let $\mathbb{R}_{\geq 0}$, $\mathbb{Z}_{\geq 0}$ be the sets of nonnegative real, integer numbers, resp., and let $\|\cdot\|$ be the Euclidean distance.

2.1 Basic geometric notions

Let $[p,q] \subset \mathbb{R}^d$ be the closed segment with extreme points p and $q \in \mathbb{R}^d$. Let $\overline{B}(p,r) = \{q \in \mathbb{R}^d \mid \|q-p\| \leq r\}$ and $H_{po} = \{q \in \mathbb{R}^d \mid \|q-p\| \leq \|q-o\|\}$ be the closed halfspace determined by $p, o \in \mathbb{R}^d$ that contains p. Let $\phi : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ be a bounded measurable function, termed *density*. For $S \subset \mathbb{R}^d$, the *mass* and *center of mass* of S with respect to ϕ are

$$M_S = \int_S \phi(q)dq, \qquad C_S = \frac{1}{M_S}\int_S q\phi(q)dq.$$

The *circumcenter* cc_S of a bounded set $S \subset \mathbb{R}^d$ is the center of the closed ball of minimum radius that contains S. The *circumradius* cr_S of S is the radius of this ball. The diameter of S is $\mathrm{diam}(S) = \max_{p,q\in S}\|p-q\|$.

Given $v \in \mathbb{R}^d \setminus \{0\}$, let $\mathrm{unit}(v)$ be the unit vector in the direction of v. For a convex set $S \subset \mathbb{R}^d$ and $p \in \mathbb{R}^d$, $\mathrm{pr}_S(p)$ is the point in S closest to p. The *to-ball-boundary* map $\mathrm{tbb} : (\mathbb{R}^d \times \mathbb{R}_{\geq 0})^2 \to \mathbb{R}^d$ takes (p,δ,q,r) to

$$\begin{cases} p + \delta\,\mathrm{unit}(q-p) & \text{if } \|p - \mathrm{pr}_{\overline{B}(q,r)}(p)\| > \delta, \\ \mathrm{pr}_{\overline{B}(q,r)}(p) & \text{if } \|p - \mathrm{pr}_{\overline{B}(q,r)}(p)\| \leq \delta. \end{cases}$$

Figure 1 illustrates the action of tbb.

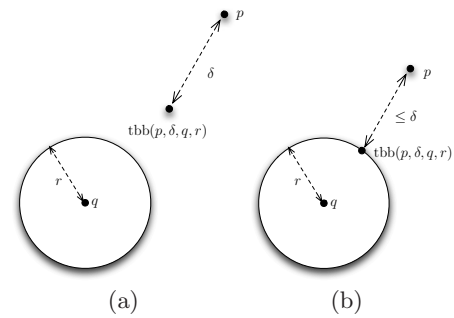


Fig. 1. Graphical representation of the action of tbb when (a) $\|p - \mathrm{pr}_{\overline{B}(q,r)}(p)\| > v_{\max}$ and (b) $\|p - \mathrm{pr}_{\overline{B}(q,r)}(p)\| \leq v_{\max}$.

The notion of Voronoi partitioning [Okabe et al., 2000] plays an important role in the later developments. Let S be a convex polygon in $\mathbb{R}^2$ including its interior and let $P = (p_1, \ldots, p_n)$ be n points in S. A *partition* of S is a collection of n polygons $\mathcal{W} = \{W_1, \ldots, W_n\}$ with disjoint interiors whose union is S. The *Voronoi partition* $\mathcal{V}(P) = \{V_1, \ldots, V_n\}$ of S generated by the points P is

$$V_i = \{q \in S \mid \|q - p_i\| \leq \|q - p_j\|\,,\ \forall j \neq i\}.$$

When the Voronoi regions V_i and V_j share an edge, p_i and p_j are *(Voronoi) neighbors*. We denote the neighbors of agent i by $\mathcal{N}_i$. $P = (p_1, \ldots, p_n)$ is a *centroidal Voronoi configuration* if $p_i = C_{V_i}$, for all $i \in \{1, \ldots, n\}$.

2.2 Facility location and aggregate distortion

We introduce here a locational optimization function termed aggregate distortion [Du et al., 1999, Bullo et al., 2009]. Consider a set of agent positions $P \in S^n$. The *agent performance* at q of the agent p_i degrades with $\|q - p_i\|^2$. Assume a density $\phi : S \to \mathbb{R}$ is available, with $\phi(q)$ reflecting the likelihood of an event happening at q. Consider then the minimization of

$$\mathcal{H}(P) = E_\phi\left[\min_{i\in\{1,\ldots,n\}} \|q - p_i\|^2\right]. \tag{1}$$

This type of function is applicable in scenarios where the agent closest to an event of interest is the one responsible for it. Examples include servicing tasks, spatial sampling of random fields, resource allocation, and event detection, see [Bullo et al., 2009] and references therein. Interestingly, $\mathcal{H}$ can be rewritten as

$$\mathcal{H}(P) = \sum_{i=1}^{n} \int_{V_i} \|q - p_i\|^2 \phi(q)dq,$$

This suggests defining a generalization of $\mathcal{H}$, which with a slight abuse of notation we denote by the same letter,

$$\mathcal{H}(P, \mathcal{W}) = \sum_{i=1}^{n} \int_{W_i} \|q - p_i\|^2 \phi(q) dq, \tag{2}$$

where $\mathcal{W}$ is a partition of S, and the ith agent is responsible of the "dominance region" W_i. The function $\mathcal{H}$ is to be minimized with respect to the locations P and the dominance regions $\mathcal{W}$. The next result [Du et al., 1999, Bullo et al., 2009] characterizes its critical points.

Lemma 2.1 *Given $P \in S^n$ and a partition $\mathcal{W}$ of S,*

$$\mathcal{H}(P, \mathcal{V}(P)) \leq \mathcal{H}(P, \mathcal{W}), \tag{3}$$

i.e., the optimal partition is the Voronoi partition. For $P' \in S^n$ with $\|p'_i - C_{W_i}\| \leq \|p_i - C_{W_i}\|$, $i \in \{1, \ldots, n\}$,

$$\mathcal{H}(P', \mathcal{W}) \leq \mathcal{H}(P, \mathcal{W}),$$

i.e., the optimal sensor positions are the centroids.

3 Problem statement

Consider a group of agents moving in a convex polygon $S \subset \mathbb{R}^2$ with positions $(p_1, \ldots, p_n)$. For simplicity, we consider arbitrary continuous-time dynamics such that

(i) all agents' clocks are synchronous, i.e., given a common starting time t_0, subsequent timesteps occur for all agents at $t_\ell = t_0 + \ell \Delta t$, for $\ell \in \mathbb{Z}_{\geq 0}$;
(ii) each agent can move at a maximum speed of $v_{\max}$, i.e., $\|p_i(t_\ell + \Delta t) - p_i(t_\ell)\| \leq v_{\max} \Delta t$;
(iii) for $p_{\text{goal}} \in S$, there exists a control such that $\|p_i(t_\ell + \Delta t) - p_{\text{goal}}\| < \|p_i(t_\ell) - p_{\text{goal}}\|$, $p_i(t_\ell + \Delta t) \in [p_i(t_\ell), p_{\text{goal}}]$ and $p_i([t_\ell, t_{\ell+1}]) \subset S$.

Later in Section 7.2, we will relax assumption (i). In our later developments, we assume in (iii) that, if $\|p_i(t_\ell) - p_{\text{goal}}\| \leq v_{\max} \Delta t$, then $p_i(t_\ell + \Delta t) = p_{\text{goal}}$ for simplicity. Dropping this assumption does not affect any results.

Our objective is to achieve optimal deployment with respect to $\mathcal{H}$, even when agents have outdated information about each others' positions. Since agents expend energy to communicate, agents need to balance the need for updated information with the desire to spend as little energy as possible. Our goal is to understand how communication effort affects deployment performance.

The data structure that each agent i maintains about other agents $j \in \{1, \ldots, n\} \setminus \{i\}$ is the last known location p^i_j and the time elapsed $\tau^i_j \in \mathbb{R}_{\geq 0}$ since this information was received (if i does not ever receive information about j, then p^i_j and τ^i_j are never initiated). For itself, agent i has access to up-to-date location information, i.e., $p^i_i = p_i$ and $\tau^i_i = 0$ at all times. When data is available, agent i knows that, at the current time, j will not have traveled more than $r^i_j = v_{\max} \tau^i_j$ from p^i_j, and hence i can construct a ball $\overline{B}(p^i_j, r^i_j)$ that is guaranteed to contain the true location of j. Note that once any radius r^i_j becomes $\text{diam}(S)$, it does not make sense to grow it any more. The data is stored in

$$\mathcal{D}^i = ((p^i_1, r^i_1), \ldots, (p^i_n, r^i_n)) \in (S \times \mathbb{R}_{\geq 0})^n.$$

Additionally, agent i maintains a set $\mathcal{A}^i \subset \{1, \ldots, n\}$ with $i \in \mathcal{A}^i$ that, at any time t, corresponds to the agents whose position information should be used. For instance, $\mathcal{A}^i = \{1, \ldots, n\}$ would mean that agent i uses all the information contained in $\mathcal{D}^i$. As we will explain in Section 5.2, this is not always necessary. We refer to $\mathcal{D} = (\mathcal{D}^1, \ldots, \mathcal{D}^n) \in (S \times \mathbb{R}_{\geq 0})^{n^2}$ as the entire memory of the network. We find it convenient to define the map loc : $(S \times \mathbb{R}_{\geq 0})^{n^2} \to S^n$, $\text{loc}(\mathcal{D}) = (p^1_1, \ldots, p^n_n)$, to extract the exact agents' location information from $\mathcal{D}$.

Remark 3.1 (Errors in position information) The model described above assumes, for simplicity, that each agent knows and transmits its own position exactly. Errors in acquiring exact information can easily be incorporated into the model if they are upper bounded by $\delta \in \mathbb{R}_{\geq 0}$ by setting $r^i_j = v_{\max} \tau^i_j + \delta$, for all $i, j \in \{1, \ldots, n\}$.•

To optimize $\mathcal{H}$, the knowledge of its own Voronoi cell is critical to each agent, cf. Section 2.2. However, with the data structure described above, agents cannot compute the Voronoi partition exactly. We address this next.

4 Space partitions with uncertain information

Since we are looking at scenarios with imperfect data, we introduce partitioning techniques with uncertainty.

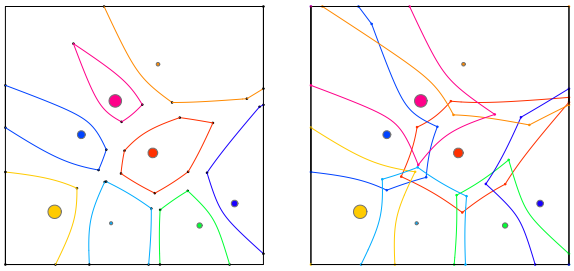
4.1 Guaranteed Voronoi diagram

Here, we follow [Sember and Evans, 2008, Jooyandeh et al., 2009]. Let $S \subset \mathbb{R}^2$ be a convex polygon and consider a set of *uncertain* regions $D_1, \ldots, D_n \subset S$, each containing a site $p_i \in D_i$. The *guaranteed Voronoi diagram* of S generated by $D = (D_1, \ldots, D_n)$ is the collection $\text{g}\mathcal{V}(D_1, \ldots, D_n) = \{\text{g}V_1, \ldots, \text{g}V_n\}$,

$$\text{g}V_i = \{q \in S \mid \max_{x \in D_i} \|q - x\| \leq \min_{y \in D_j} \|q - y\| \text{ for all } j \neq i\}.$$

With a slight abuse of notation, $\text{g}V_i(D)$ denotes the ith component of $\text{g}\mathcal{V}(D_1, \ldots, D_n)$. Note that $\text{g}V_i$ contains the points of S that are guaranteed to be closer to p_i than to any other of the nodes p_j, $j \neq i$. Because the information about the location of these nodes is uncertain, there is a neutral region in S which is not assigned to anybody: those points for which no guarantee can be established. The guaranteed Voronoi diagram is not a partition of S, see Figure 2(a). Each point in the boundary of $\text{g}V_i$ belongs to the set

$$\Delta^{\text{g}}_{ij} = \{q \in S \mid \max_{x \in D_i} \|q - x\| = \min_{y \in D_j} \|q - y\|\}, \tag{4}$$



(a) (b)

Fig. 2. Guaranteed (a) and dual guaranteed (b) Voronoi diagrams.

for some $j \neq i$. Note that $\Delta^{\mathrm{g}}_{ij} \neq \Delta^{\mathrm{g}}_{ji}$. If every region D_i is a point, $D_i = \{p_i\}$, then $\mathrm{g}\mathcal{V}(D_1, \ldots, D_n) = \mathcal{V}(p_1, \ldots, p_n)$. For any collection of points $p_i \in D_i$, $i \in \{1, \ldots, n\}$, the guaranteed Voronoi diagram is contained in the Voronoi partition, i.e., $\mathrm{g}V_i \subset V_i$, $i \in \{1, \ldots, n\}$. Agent p_j is a guaranteed Voronoi neighbor of p_i if $\Delta^{\mathrm{g}}_{ij} \cap \partial \mathrm{g}V_i$ is not empty nor a singleton. The set of guaranteed Voronoi neighbors of agent i is $\mathrm{g}\mathcal{N}_i(D)$.

Throughout the paper, we consider uncertain regions given by balls, $D_i = \overline{B}(p_i, r_i)$, $i \in \{1, \ldots, n\}$. Then, the edges (4) composing the boundary of $\mathrm{g}V_i$ are given by

$$\Delta^{\mathrm{g}}_{ij} = \{q \in S \mid \|q - p_i\| + r_i = \|q - p_j\| - r_j\}, \quad (5)$$

thus they lie on the arm of the hyperbola closest to p_i with foci p_i and p_j, and semimajor axis $\frac{1}{2}(r_i + r_j)$. Note that each cell is convex. The following results states a useful property of the guaranteed Voronoi diagram.

Lemma 4.1 *Given $p_1, \ldots, p_n \in S$ and $r_1, \ldots, r_n, a \in \mathbb{R}_{\geq 0}$, let $D_i = \overline{B}(p_i, r_i)$ and $D'_i = \overline{B}(p_i, r_i + a)$, for $i \in \{1, \ldots, n\}$. Then, $g\mathcal{N}_i(D'_1, \ldots, D'_n) \subset g\mathcal{N}_i(D_1, \ldots, D_n)$, for all $i \in \{1, \ldots, n\}$.*

PROOF. Let $j \in \mathrm{g}\mathcal{N}_i(D')$. This fact implies, according to (5), that there exists $q \in S$ such that

$$\begin{aligned} \|q - p_i\| + r_i + a &= \|q - p_j\| - r_j - a \\ &< \|q - p_k\| - r_k - a, \end{aligned} \quad (6)$$

for all $k \in \{1, \ldots, n\} \setminus \{i, j\}$. Now, let q' be the unique point in $[q, p_j]$ such that $\|q' - p_i\| + r_i = \|q' - p_j\| - r_j$. Note that, since $q' \in [q, p_j]$, then $\|q' - p_j\| = \|q - p_j\| - \|q' - q\|$. Therefore, we can write

$$\begin{aligned} \|q' - p_j\| - r_j &= \|q - p_j\| - r_j - \|q' - q\| \\ &< \|q - p_k\| - r_k - \|q' - q\|, \end{aligned}$$

for all $k \in \{1, \ldots, n\} \setminus \{i, j\}$, where we have used (6). Now, using the triangle inequality $\|q - p_k\| \leq \|q - q'\| + \|q' - p_k\|$, we deduce $\|q' - p_j\| - r_j < \|q' - p_k\| - r_k$, for all $k \in \{1, \ldots, n\} \setminus \{i, j\}$, and hence $j \in \mathrm{g}\mathcal{N}_i(D)$. □

4.2 Dual guaranteed Voronoi diagram

Here we introduce the concept of dual guaranteed Voronoi diagram. We first define a *covering* of Q as a collection of n polytopes $\mathcal{W} = \{W_1, \ldots, W_n\}$ whose union is Q but do not necessarily have disjoint interiors. The *dual guaranteed Voronoi diagram* of S generated by $D_1, \ldots, D_n$ is the collection of sets $\mathrm{dg}\mathcal{V}(D_1, \ldots, D_n) = \{\mathrm{dg}V_1, \ldots, \mathrm{dg}V_n\}$ defined by

$$\mathrm{dg}V_i = \{q \in S \mid \min_{x \in D_i} \|q - x\| \leq \max_{y \in D_j} \|q - y\| \text{ for all } j \neq i\}.$$

With a slight abuse of notation, $\mathrm{dg}V_i(D)$ denotes the ith component of $\mathrm{dg}\mathcal{V}(D_1, \ldots, D_n)$. Note that the points of S outside $\mathrm{dg}V_i$ are guaranteed to be closer to some other node p_j, $j \neq i$ than to p_i. Because the information about the location of these nodes is uncertain, there are regions of the space that belong to more than one cell. The dual guaranteed Voronoi diagram is a covering of the set S, see Figure 2(b). Each point in the boundary of $\mathrm{dg}V_i$ belongs to the set

$$\Delta^{\mathrm{dg}}_{ij} = \{q \in S \mid \min_{x \in D_i} \|q - x\| = \max_{y \in D_j} \|q - y\|\}, \quad (7)$$

for some $j \neq i$. Note that $\Delta^{\mathrm{dg}}_{ij} \neq \Delta^{\mathrm{dg}}_{ji}$. If every region D_i is a point, $D_i = \{p_i\}$, then $\mathrm{dg}\mathcal{V}(D_1, \ldots, D_n) = \mathcal{V}(p_1, \ldots, p_n)$. For any collection of points $p_i \in D_i$, $i \in \{1, \ldots, n\}$, the guaranteed Voronoi covering contains the Voronoi partition, i.e., $V_i \subset \mathrm{dg}V_i$, $i \in \{1, \ldots, n\}$. Agent p_j is a dual guaranteed Voronoi neighbor of p_i if $\Delta^{\mathrm{dg}}_{ij} \cap \partial \mathrm{dg}V_i$ is not empty nor a singleton. The set of dual guaranteed Voronoi neighbors of i is $\mathrm{dg}\mathcal{N}_i(D)$.

Consider the uncertain regions given $D_i = \overline{B}(p_i, r_i)$, $i \in \{1, \ldots, n\}$. The edges (7) are given by

$$\Delta^{\mathrm{dg}}_{ij} = \{q \in S \mid \|q - p_i\| - r_i = \|q - p_j\| + r_j\}, \quad (8)$$

thus they lie on the arm of the hyperbola farthest from p_i with foci p_i and p_j, and semimajor axis $\frac{1}{2}(r_i + r_j)$. Cells are generally not convex. The next result states a useful property of the dual guaranteed Voronoi diagram. Its proof is analogous to that of Lemma 4.1 and is omitted.

Lemma 4.2 *Given $p_1, \ldots, p_n \in S$ and $r_1, \ldots, r_n, a \in \mathbb{R}_{\geq 0}$, let $D_i = \overline{B}(p_i, r_i)$ and $D'_i = \overline{B}(p_i, r_i + a)$, for $i \in \{1, \ldots, n\}$. Then, $dg\mathcal{N}_i(D_1, \ldots, D_n) \subset dg\mathcal{N}_i(D'_1, \ldots, D'_n)$, for all $i \in \{1, \ldots, n\}$.*

The next result is another useful property. Its proof is a direct consequence of the definition of $\mathrm{dg}V$.

Lemma 4.3 *Given sets $D_1, \ldots, D_{n+m} \subset S$, it holds that $dgV_i(D_1, \ldots, D_n, D_{n+1}, \ldots, D_{n+m}) \subseteq dgV_i(D_1, \ldots, D_n)$ for all $i \in \{1, \ldots, n\}$.*

5 Self-triggered coverage optimization

Here we design an algorithm to solve the problem described in Section 3. From the point of view of an agent, the algorithm is composed of two components: a motion control part that determines the best way to move given the available information and an update decision part that determines when new information is needed.

5.1 Motion control

If an agent had perfect knowledge of other agents' positions, then to optimize $\mathcal{H}$, it could compute its own Voronoi cell and move towards its centroid, as in [Cortés et al., 2004]. Since this is not the case we are considering, we instead propose an alternative motion control law. Let us describe it first informally:

> *[Informal description]:* At each timestep, each agent uses its stored information about other agents' locations to calculate its own guaranteed Voronoi and dual guaranteed Voronoi cells. Then, the agent moves towards the centroid of its guaranteed Voronoi cell.

Note that this law assumes that each agent has access to the value of the density ϕ over its guaranteed Voronoi cell. In general, there is no guarantee that following the `motion control law` will lead the agent to get closer to the centroid of its Voronoi cell. A condition under which this statement holds is characterized by the next result.

Lemma 5.1 *Given $p \neq q, q^* \in \mathbb{R}^2$, let $p' \in [p, q]$ such that $\|p' - q\| \geq \|q^* - q\|$. Then, $\|p' - q^*\| \leq \|p - q^*\|$.*

PROOF. We reason by contradiction. Assume $\|p' - q^*\| > \|p - q^*\|$. Since $p' \in [p, q]$, we have $\angle(q - p', q^* - p') = \pi - \angle(p - p', q^* - p')$. Now,

$$\begin{aligned}(p - p') \cdot (q^* - p') &= (p - q^* + q^* - p') \cdot (q^* - p') \\ &= (p - q^*) \cdot (q^* - p') + \|q^* - p'\|^2.\end{aligned}$$

Since $\|p' - q^*\| > \|p - q^*\|$, it follows that $\angle(p - p', q^* - p') \in [0, \pi/2)$, and hence $\angle(q - p', q^* - p') \in (\pi/2, \pi]$. Now the application of the law of cosines to the triangle with vertices q^*, q, and p' yields

$$\begin{aligned}\|q^* - q\|^2 &= \|q^* - p'\|^2 + \|q - p'\|^2 \\ &- 2\|q^* - p'\|\|q - p'\| \cos\angle(q - p', q^* - p') > \|q - p'\|^2,\end{aligned} \tag{9}$$

where we use the fact that $p' \neq q^*$ (otherwise, $\|p' - q^*\| > \|p - q^*\|$ would imply $p = q^*$, a contradiction). Finally, the result follows by noting that (9) contradicts the hypothesis $\|p' - q\| \geq \|q^* - q\|$. □

Hence, with the notation of Lemma 5.1, if i is at $p = p_i$, computes the goal $q = C_{\mathrm{g}V_i}$ and moves towards it to p', then the distance to $q^* = C_{V_i}$ decreases as long as

$$\|p' - C_{\mathrm{g}V_i}\| \geq \|C_{V_i} - C_{\mathrm{g}V_i}\| \tag{10}$$

holds. This is illustrated in Figure 3. The right-hand side cannot be computed by i because of lack of information about C_{V_i} but can be upper bounded, as we show next.

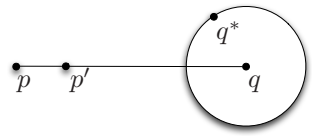


Fig. 3. Graphical representation of Lemma 5.1.

Proposition 5.2 *Let $L \subset V \subset U$. Then, for any density function ϕ, the following holds*

$$\|C_V - C_L\| \leq 2cr_U\Big(1 - \frac{M_L}{M_U}\Big). \tag{11}$$

PROOF. For convenience, let $a = M_L$, $b = M_V$, and $c = M_U$. By hypothesis, $a \leq b \leq c$. Note also that $M_{U \setminus L} = c - a$. By definition, we have

$$C_V - C_L = \frac{1}{b}\int_V q\phi(q)dq - \frac{1}{a}\int_L q\phi(q)dq. \tag{12}$$

For any $v \in \mathbb{R}^2$, $v = \frac{1}{b}\int_V v\phi(q)dq = \frac{1}{a}\int_L v\phi(q)dq$. Summing and subtracting $v \in \mathbb{R}^2$, we get that (12) equals

$$\begin{aligned}&\frac{1}{b}\int_V (q - v)\phi(q)dq - \frac{1}{a}\int_L (q - v)\phi(q)dq \\ =&\frac{1}{b}\int_{V \setminus L} (q - v)\phi(q)dq + \Big(\frac{1}{b} - \frac{1}{a}\Big)\int_L (q - v)\phi(q)dq.\end{aligned}$$

Taking norms, we deduce that for $v = \mathrm{cc}_U$, we have

$$\|C_V - C_L\| \leq \frac{1}{b}\mathrm{cr}_U(b - a) + \Big|\frac{1}{b} - \frac{1}{a}\Big|\mathrm{cr}_U a.$$

The result now follows after some manipulations. □

In general, the bound in Proposition 5.2 is tight, i.e., there exist density functions for which (11) is an equality. Exploiting Proposition 5.2, agent i can use $L = \mathrm{g}V_i$ and $U = \mathrm{dg}V_i$ to upper bound the distance $\|C_{V_i} - C_{\mathrm{g}V_i}\|$ by

$$\mathrm{bnd}_i \equiv \mathrm{bnd}(\mathrm{g}V_i, \mathrm{dg}V_i) = 2\mathrm{cr}_{\mathrm{dg}V_i}\Big(1 - \frac{M_{\mathrm{g}V_i}}{M_{\mathrm{dg}V_i}}\Big). \tag{13}$$

This bound is computable with information in $\mathcal{D}^i$ only and can be used to guarantee that (10) holds by ensuring

$$\|p' - C_{\mathrm{g}V_i}\| \geq \mathrm{bnd}_i \tag{14}$$

holds. The point p' to which agent i moves to is determined as follows: move towards $C_{\mathrm{g}V_i}$ as much as possible

Agent $i \in \{1, \ldots, n\}$ performs:

1: set $D = \mathcal{D}^i$
2: compute $L = \mathrm{gV}_i(D)$ and $U = \mathrm{dgV}_i(D)$
3: compute $q = C_L$ and $r = \mathrm{bnd}(L, U)$
4: move to $\mathrm{tbb}(p_i, v_{\max}\Delta t, q, r)$
5: set $\mathcal{D}^i_j = (p^i_j, \min\{r^i_j + v_{\max}\Delta t, \mathrm{diam}(S)\}$
6: set $\mathcal{D}^i_i = (\mathrm{tbb}(p_i, v_{\max}\Delta t, q, r), 0)$

Table 1
`motion control law`.

in one time step until it is within distance bnd_i of it. Formally, the `motion control law` is described in Table 1.

If time elapses without new location information, then the uncertainty radii in the agent memory's grows, the bound (13) grows larger and (14) becomes harder to satisfy until it becomes unfeasible. Therefore, agents need a decision mechanism that establishes when new information is required in order for the execution of the motion control law to be useful. This is addressed in Section 5.2.

5.2 Update decision policy

The second component of our coordination strategy takes care of updating the memory of the agents, and in particular, of deciding when new information is needed. To specify this component, we build on the discussion of the previous section, specifically on making sure that condition (14) is feasible. Two reasons can make this condition invalid for a given agent i. One reason is the bound bnd_i might be large due to outdated location information about other agents' location in $\mathcal{D}^i$. This should trigger the need for up-to-date information through communication with other agents. Another reason is that agent i might be close to C_{gV_i}, requiring bnd_i to be very small. We deal with this by specifying a tolerance $\varepsilon > 0$ that is selected a priori by the designer.

We describe the decision policy informally next.

> *[Informal description]:* At each timestep, each agent uses its stored information about other agents' locations to calculate its own guaranteed Voronoi and dual guaranteed Voronoi cells, and the bound (13). Then, it decides that up-to-date location information is required if its computed bound is larger than ε and the distance to the centroid of its guaranteed cell.

Formally, the memory updating mechanism followed by each agent is described by the pseudo-code in Table 2.

Agent $i \in \{1, \ldots, n\}$ performs:

1: set $D = \mathcal{D}^i$
2: compute $L = \mathrm{gV}_i(D)$ and $U = \mathrm{dgV}_i(D)$
3: compute $q = C_L$ and $r = \mathrm{bnd}(L, U)$
4: **if** $r \geq \max\{\|q - p_i\|, \varepsilon\}$ **then**
5: reset $\mathcal{D}^i$ by acquiring up-to-date location information
6: **end if**

Table 2
`one-step-ahead update decision policy`.

According to Table 2, agent i checks at each time step if condition (14) is feasible or $\mathrm{bnd}_i \leq \varepsilon$, and therefore it is advantageous to execute the `motion control law`. An equivalent way of describing this decision policy that more clearly displays its self-triggered character is given by the `multiple-steps-ahead update decision policy` of Table 3. According to Table 3, agent i determines when in the future it will have to update its location information again as a function of the current state of its memory.

Agent $i \in \{1, \ldots, n\}$ performs:

1: set $D = \mathcal{D}^i$
2: compute $L = \mathrm{gV}_i(D)$ and $U = \mathrm{dgV}_i(D)$
3: compute $q = C_L$ and $r = \mathrm{bnd}(L, U)$
4: **if** $r \geq \max\{\|q - p_i\|, \varepsilon\}$ **then**
5: reset $\mathcal{D}^i$ by acquiring up-to-date location information
6: **else**
7: initialize $t_{\mathrm{sleep}} = 0$
8: **while** $r < \max\{\|q - p_i\|, \varepsilon\}$ **do**
9: set $t_{\mathrm{sleep}} = t_{\mathrm{sleep}} + 1$
10: set $\mathcal{D}^i_j = (p^i_j, \min\{r^i_j + v_{\max}\Delta t, \mathrm{diam}(S)\})$ for $j \neq i$
11: set $\mathcal{D}^i_i = (\mathrm{tbb}(p_i, v_{\max}\Delta t, q, r), 0)$
12: set $D = \mathcal{D}^i$
13: compute $L = \mathrm{gV}_i(D)$ and $U = \mathrm{dgV}_i(D)$
14: compute $q = C_L$ and $r = \mathrm{bnd}(L, U)$
15: **end while**
16: execute policy again in t_{sleep} timesteps
17: **end if**

Table 3
`multiple-steps-ahead update decision policy`.

5.3 The `self-triggered centroid algorithm`

Here, we synthesize a self-triggered algorithm to achieve optimal deployment with outdated information. The algorithm is the result of combining the motion control law of Section 5.1 and the update decision policies of Section 5.2 with a procedure to acquire up-to-date information about other agents when this requirement is triggered (cf. `5:` in both Tables 2 and 3). Let us discuss this latter point in detail. A trivial update mechanism will be to provide each agent with up-to-date information about the location of everybody else in the network. However, the implementation of such a mechanism is costly from a communications point of view. We instead propose to use an alternative algorithm that only provides up-to-date location information of the Voronoi neighbors at the specific time when step `5:` is executed. This algorithm, termed the `Voronoi cell computation`, is borrowed from [Cortés et al., 2004]. We present it in Table 4, adapted to our scenario.

The `Voronoi cell computation` determines a radius R_i with the property that agent i does not need location information about agents farther away than R_i from p_i to compute exactly its Voronoi cell. There are multiple ways as to how an agent might physically acquire location information about agents located within a distance

At timestep $\ell \in \mathbb{Z}_{\geq 0}$, agent $i \in \{1, \ldots, n\}$ performs:

initialize $R_i = \min_{k \in \{1,\ldots,n\} \setminus \{i\}} \|p_i - p_k^i\| + v_{\max} \tau_k^i$
detect all p_j within radius R_i
set $W(p_i, R_i) = \overline{B}(p_i, R_i) \cap \big(\cap_{j : \|p_i - p_j\| \leq R_i} H_{p_i p_j} \big)$
while $R_i < 2 \max_{q \in W(p_i, R_i)} \|p_i - q\|$ **do**
  set $R_i := 2R_i$
  detect all p_j within radius R_i
  set $W(p_i, R_i) = \overline{B}(p_i, R_i) \cap \big(\cap_{j : \|p_i - p_j\| \leq R_i} H_{p_i p_j} \big)$
end while
set $V_i = W(p_i, R_i)$
set $\mathcal{A}^i = \mathcal{N}_i \cup \{i\}$ and $\mathcal{D}_j^i = (p_j, 0)$ for $j \in \mathcal{N}_i$

Table 4
`Voronoi cell computation`.

less than or equal to this radius, including point-to-point communication, multi-hop communication, and sensing.

The next result justifies why an agent i may use only the subset $\mathcal{A}^i$ prescribed by `Voronoi cell computation` to compute L and U in the algorithms presented above. In the statement, $\pi_{\mathcal{A}^i}$ denotes the map that extracts from $\mathcal{D}^i$ the information about the agents contained in $\mathcal{A}^i$.

Lemma 5.3 *Assume that at timestep $\ell_* \in \mathbb{Z}_{\geq 0}$, agent $i \in \{1, \ldots, n\}$ gets up-to-date information about the location of its current Voronoi neighbors (e.g., by executing the* `Voronoi cell computation`*). Let $D_{all}(\ell_*) = ((p_1(t_{\ell_*}), 0), \ldots, (p_n(t_{\ell_*}), 0)) \in (S \times \mathbb{R}_{\geq 0})^n$ and let $D_{Vr}(\ell_*) \in (S \times \mathbb{R}_{\geq 0})^n$ be any vector whose jth component is $(p_j(t_{\ell_*}), 0)$, for all $j \in \mathcal{A}^i = \mathcal{N}_i \cup \{i\}$. For $\ell \geq \ell_*$, define recursively*

$$L_{Vr}(\ell) = gV(\pi_{\mathcal{A}^i}(D_{Vr}(\ell))), \quad U_{Vr}(\ell) = dgV(\pi_{\mathcal{A}^i}(D_{Vr}(\ell))),$$
$$L_{all}(\ell) = gV(D_{all}(\ell)), \qquad U_{all}(\ell) = dgV(D_{all}(\ell)),$$

where $D_{Vr}(\ell + 1) = \mathrm{Evl}_\ell(D_{Vr}(\ell))$, $D_{all}(\ell + 1) = \mathrm{Evl}_\ell(D_{all}(\ell))$ and $\mathrm{Evl}_\ell : (S \times \mathbb{R}_{\geq 0})^n \to (S \times \mathbb{R}_{\geq 0})^n$, corresponding to the time evolution of the data structure, is given by $(\mathrm{Evl}_\ell)_j(D) = (p_j, r_j + v_{max}\Delta t)$ for $j \neq i$, and

$$(\mathrm{Evl}_\ell)_i(D) = (\mathrm{tbb}(p_i, v_{max}, C_{L_{Vr}(\ell)}, \mathrm{bnd}(L_{Vr}(\ell), U_{Vr}(\ell))), 0),$$

otherwise. Then, for $\ell \geq \ell_*$,

$$L_{Vr}(\ell) = L_{all}(\ell) \text{ and } U_{All}(\ell) \subset U_{Vr}(\ell).$$

Lemma 5.3 states that the information provided by the `Voronoi cell computation` is sufficient to compute the quantities required by the motion control law and the update decision policies. Its proof follows from Lemmas 4.1 and 4.3. Lemma 4.1 implies that taking into account only the uncertain positions of agents in $\mathcal{A}^i$ is enough to compute correctly the guaranteed Voronoi cell. Lemma 4.3 implies that using only this information an upper bound of the dual guaranteed Voronoi cell can be computed. Thus the `self-triggered centroid algorithm` can be run by agent i using only the information in $\pi_{\mathcal{A}^i}(\mathcal{D}^i)$.

The combination of the `Voronoi cell computation` with the `motion control law`, cf. Section 5.1, and the `one-step-ahead update decision policy`, cf. Section 5.2, leads to the synthesis of the `self-triggered centroid algorithm` in Table 5. A similar version of this algorithm can be written using the `multiple-steps-ahead update decision policy` in which agents can instead schedule the next time information should be updated as opposed to checking condition (14) in each intermediate timestep. Since the latter corresponds to multiple executions of the `one-step-ahead update decision policy`, the trajectories described by the network would be the same, and hence we just concentrate on the analysis of the `self-triggered centroid algorithm`.

Initialization

set $\mathcal{D}^i$ and $\mathcal{A}^i$ by running `Voronoi cell computation`
set $\mathcal{D}_i^i = (p_i, 0)$

At timestep $\ell \in \mathbb{Z}_{\geq 0}$, agent $i \in \{1, \ldots, n\}$ performs:

set $D = \pi_{\mathcal{A}^i}(\mathcal{D}^i)$
compute $L = gV_i(D)$ and $U = dgV_i(D)$
compute $q = C_L$ and $r = \mathrm{bnd}(L, U)$
if $r \geq \max\{\|q - p_i\|, \varepsilon\}$ **then**
  reset $\mathcal{D}^i$ and $\mathcal{A}^i$ by running `Voronoi cell computation`
  set $D = \pi_{\mathcal{A}^i}(\mathcal{D}^i)$
  set $L = gV(D)$ and $U = dgV(D)$
  set $q = C_L$ and $r = \mathrm{bnd}(L, U)$
end if
move to $\mathrm{tbb}(p_i, v_{\max}\Delta t, q, r)$
set $\mathcal{D}_i^i = (\mathrm{tbb}(p_i, v_{\max}\Delta t, q, r), 0)$
set $\mathcal{D}_j^i = (p_j^i, \min\{r_j^i + v_{\max}\Delta t, \mathrm{diam}(S)\})$ for $j \in \mathcal{A}^i \setminus \{i\}$

Table 5
`self-triggered centroid algorithm`.

Remark 5.4 (Robustness against agent departures and arrivals) The `self-triggered centroid algorithm` is robust against agent departures and arrivals. Consider the case of a failing agent i that can no longer send or receive information to/from any other agent j. Once all other agents j have updated their information according to `Voronoi cell computation`, notice that $i \notin \mathcal{A}^j$ for all the remaining agents j, which continue to run the `self-triggered centroid algorithm` normally without agent i. On the other hand if a new agent i appears in the system, we require it to immediately update its information and send a request to its Voronoi neighbors to do the same thing. After this, the `self-triggered centroid algorithm` can continue running having incorporated agent i. •

6 Convergence of synchronous executions

In this section, we analyze the asymptotic convergence properties of the `self-triggered centroid algorithm`. Note that this algorithm can be written

as a map $f_{\text{stca}} : (S \times \mathbb{R}_{\geq 0})^{n^2} \to (S \times \mathbb{R}_{\geq 0})^{n^2}$ which corresponds to the composition of a "decide/acquire-up-to-date-information" map f_{info} and a "move-and-update-uncertainty" map f_{motion}, i.e., $f_{\text{stca}}(\mathcal{D}) = f_{\text{motion}}(f_{\text{info}}(\mathcal{D}))$ for $\mathcal{D} \in (S \times \mathbb{R}_{\geq 0})^{n^2}$. Our analysis strategy here is shaped by the fact that f_{info}, and consequently, f_{stca} are discontinuous.

Our objective is to prove the following result characterizing the asymptotic convergence properties of the trajectories of the `self-triggered centroid algorithm`.

Proposition 6.1 *For $\varepsilon \in [0, \text{diam}(S))$, the agents' position evolving under the* ***`self-triggered centroid algorithm`*** *from any initial network configuration in S^n converges to the set of centroidal Voronoi configurations.*

Since the map f_{stca} is discontinuous, we cannot readily apply the discrete-time LaSalle Invariance Principle. Our strategy to prove Proposition 6.1 is to construct a closed set-valued map T_{sync}, whose trajectories include the ones of f_{stca}, and apply the LaSalle Invariance Principle for set-valued maps, e.g., [Bullo et al., 2009].

Next, we define T_{sync} formally. For convenience, we recall that $\mathcal{D} = (\mathcal{D}^1, \ldots, \mathcal{D}^n) \in (S \times \mathbb{R}_{\geq 0})^{n^2}$, and that the elements of $\mathcal{D}^i$ are referred to as $((p_1^i, r_1^i), \ldots, (p_n^i, r_n^i))$, for each $i \in \{1, \ldots, n\}$. To ease the exposition, we divide the construction of T_{sync} in two steps, a first one that captures the agent motion and the uncertainty update to the network memory, and a second one that captures the acquisition of up-to-date network information.

Motion and uncertainty update. We define the continuous motion and time update map as $\mathcal{M} : (S \times \mathbb{R}_{\geq 0})^{n^2} \to (S \times \mathbb{R}_{\geq 0})^{n^2}$ whose ith component is

$$\begin{aligned}\mathcal{M}_i(\mathcal{D}) = \big((p_1^i, \min\{r_1^i + v_{\max}\Delta t, \text{diam}(S)\}), \ldots, \\ (\text{tbb}(p_i^i, v_{\max}, C_{\text{gV}_i}(\pi_{\mathcal{A}^i}(\mathcal{D}^i)), \text{bnd}(\pi_{\mathcal{A}^i}(\mathcal{D}^i))), 0), \\ \ldots, (p_n^i, \min\{r_n^i + v_{\max}\Delta t, \text{diam}(S)\})\big),\end{aligned}$$

where $\mathcal{A}^i = \{i\} \cup \text{argmin}_{j \in \{1,\ldots,n\} \setminus \{i\}} r_j^i$.

Acquisition of up-to-date information. At each possible state of the network memory, agents are faced with the decision of whether to acquire up-to-date information about the location of other agents. This is captured by the set-valued map $\mathcal{U} : (S \times \mathbb{R}_{\geq 0})^{n^2} \rightrightarrows (S \times \mathbb{R}_{\geq 0})^{n^2}$ that, to $\mathcal{D} \in (S \times \mathbb{R}_{\geq 0})^{n^2}$, associates the Cartesian product $\mathcal{U}(\mathcal{D})$ whose ith component is either $\mathcal{D}^i$ (agent i does not get any up-to-date information) or the vector

$$((p_1', r_1'), \ldots, (p_n', r_n'))$$

where $(p_j', r_j') = (p_j^j, 0)$ for $j \in \{i\} \cup \mathcal{N}_i$ and $(p_j', r_j') = (p_j^i, r_j^i)$ otherwise (agent i gets updated information). Recall that $\mathcal{N}_i$ is the set of neighbors of agent i given the partition $\mathcal{V}(\text{loc}(\mathcal{D}))$. It is not difficult to show that $\mathcal{U}$ is closed (a set-valued map $T : X \rightrightarrows Y$ is closed if $x_k \to x$, $y_k \to y$ and $y_k \in T(x_k)$ imply that $y \in T(x)$).

We define the set-valued map $T_{\text{sync}} : (S \times \mathbb{R}_{\geq 0})^{n^2} \rightrightarrows (S \times \mathbb{R}_{\geq 0})^{n^2}$ by $T_{\text{sync}} = \mathcal{U} \circ \mathcal{M}$. Given the continuity of $\mathcal{M}$ and the closedness of $\mathcal{U}$, the map T_{sync} is closed. Moreover, if $\gamma = \{\mathcal{D}(t_\ell)\}_{\ell \in \mathbb{Z}_{\geq 0}}$ is an evolution of the `self-triggered centroid algorithm`, then $\gamma' = \{\mathcal{D}'(t_\ell)\}_{\ell \in \mathbb{Z}_{\geq 0}}$, with $\mathcal{D}'(t_\ell) = f_{\text{info}}(\mathcal{D}(t_\ell))$, is a trajectory of

$$\mathcal{D}'(t_{\ell+1}) \in T_{\text{sync}}(\mathcal{D}'(t_\ell)). \tag{15}$$

The next result establishes the monotonic evolution of the aggregate function $\mathcal{H}$ along the trajectories of T_{sync}. With a slight abuse of notation, denote also by $\mathcal{H}$ the extension of the aggregate function to the space $(S \times \mathbb{R}_{\geq 0})^{n^2}$, i.e., $\mathcal{H}(\mathcal{D}) = \mathcal{H}(\text{loc}(\mathcal{D}))$, for $\mathcal{D} \in (S \times \mathbb{R}_{\geq 0})^{n^2}$.

Lemma 6.2 *$\mathcal{H} : (S \times \mathbb{R}_{\geq 0})^{n^2} \to \mathbb{R}$ is monotonically nonincreasing along the trajectories of T_{sync}.*

PROOF. Let $\mathcal{D} \in (S \times \mathbb{R}_{\geq 0})^{n^2}$ and $\mathcal{D}' \in T_{\text{sync}}(\mathcal{D})$. For convenience, let $P = \text{loc}(\mathcal{D})$ and $P' = \text{loc}(\mathcal{D}') = \text{loc}(\mathcal{M}(\mathcal{D}))$. To establish $\mathcal{H}(P') \leq \mathcal{H}(P)$, we use the formulation (2) and divide our reasoning in two steps. First, we fix the partition $\mathcal{V}(P)$. For each $i \in \{1, \ldots, n\}$, if $\|p_i^i - C_{\text{gV}_i}(\pi_{\mathcal{A}^i}(\mathcal{D}^i))\| \leq \text{bnd}(\pi_{\mathcal{A}^i}(\mathcal{D}^i))$, then $p_i'^i = p_i^i$ because agent i does not move according to the definition of tbb. If, instead, $\|p_i^i - C_{\text{gV}_i}(\pi_{\mathcal{A}^i}(\mathcal{D}^i))\| > \text{bnd}(\pi_{\mathcal{A}^i}(\mathcal{D}^i))$, then, by Lemma 5.1 and Proposition 5.2, we have that $\|p_i'^i - C_{V_i}\| < \|p_i^i - C_{V_i}\|$. In either case, it follows from Lemma 2.1 that $\mathcal{H}(P', \mathcal{V}(P)) \leq \mathcal{H}(P, \mathcal{V}(P))$. Second, the optimality of the Voronoi partition stated in Lemma 2.1 guarantees that $\mathcal{H}(P', \mathcal{V}(P')) \leq \mathcal{H}(P', \mathcal{V}(P))$, and the result follows. □

One can establish the next result using Lemma 6.2 and the fact that T_{sync} is closed and its trajectories are bounded and belong to the closed set $(S \times \mathbb{R}_{\geq 0})^{n^2}$.

Lemma 6.3 *Let γ' be a trajectory of* (15)*. Then, the ω-limit set $\emptyset \neq \Omega(\gamma') \subset (S \times \mathbb{R}_{\geq 0})^{n^2}$ belongs to $\mathcal{H}^{-1}(c)$, for some $c \in \mathbb{R}$, and is weakly positively invariant for T_{sync}, i.e., for $\mathcal{D} \in \Omega(\gamma')$, $\exists\, \mathcal{D}' \in T_{\text{sync}}(\mathcal{D})$ with $\mathcal{D}' \in \Omega(\gamma')$.*

PROOF. Let γ' be a trajectory of (15). The fact that $\Omega(\gamma') \neq \emptyset$ follows from γ' being bounded. Let $\mathcal{D}' \in \Omega(\gamma')$. Then there exists a subsequence $\{\mathcal{D}'(t_{\ell_m}) \mid m \in \mathbb{Z}_{\geq 0}\}$ of γ' such that $\lim_{m \to +\infty} \mathcal{D}'(t_{\ell_m}) = \mathcal{D}'$. Consider $\{\mathcal{D}'(t_{\ell_{m+1}}) \mid m \in \mathbb{Z}_{\geq 0}\}$. Since this sequence is bounded, it must have a convergent subsequence, i.e., there exists $\widehat{\mathcal{D}}'$ such that $\lim_{m \to +\infty} \mathcal{D}'(t_{\ell_{m+1}}) = \widehat{\mathcal{D}}'$. By definition, $\widehat{\mathcal{D}}' \in \Omega(\gamma')$. Also, since T_{sync} is closed, we have $\widehat{\mathcal{D}}' \in T_{\text{sync}}(\mathcal{D}')$, which implies that $\Omega(\gamma')$ is weakly positively invariant. Now consider the sequence $\mathcal{H} \circ \gamma = \{\mathcal{H}(\gamma(l)) \mid l \in \mathbb{Z}_{\geq 0}\}$. Since γ is bounded and $\mathcal{H}$

is non-increasing along γ on W, the sequence $\mathcal{H}\circ\gamma$ is decreasing and bounded from below, and therefore, convergent. Let $c \in \mathbb{R}$ satisfy $\lim_{l\to+\infty}\mathcal{H}(\gamma(l)) = c$. Next, we prove that the value of V on $\Omega(\gamma)$ is constant and equal to c. Take any $z \in \Omega(\gamma)$. Accordingly, there exists a subsequence $\{\gamma(l_m) \mid m \in \mathbb{Z}_{\geq 0}\}$ such that $\lim_{m\to+\infty}\gamma(l_m) = z$. Since $\mathcal{H}$ is continuous, $\lim_{m\to+\infty}\mathcal{H}(\gamma(l_m)) = \mathcal{H}(z)$. From $\lim_{l\to+\infty}\mathcal{H}(\gamma(l)) = c$, we conclude $\mathcal{H}(z) = c$. □

We are finally ready to establish the asymptotic convergence of the `self-triggered centroid algorithm`.

PROOF OF PROPOSITION 6.1. Let $\gamma = \{\mathcal{D}(t_\ell)\}_{\ell\in\mathbb{Z}_{\geq 0}}$ be an evolution of `self-triggered centroid algorithm`. Define $\gamma' = \{\mathcal{D}'(t_\ell)\}_{\ell\in\mathbb{Z}_{\geq 0}}$ by $\mathcal{D}'(t_\ell) = f_{\text{info}}(\mathcal{D}(t_\ell))$. Note that $\text{loc}(\mathcal{D}(t_\ell)) = \text{loc}(\mathcal{D}'(t_\ell))$. Since γ' is a trajectory of T_{sync}, Lemma 6.3 guarantees that $\Omega(\gamma')$ is weakly positively invariant and belongs to $\mathcal{H}^{-1}(c)$, for some $c \in \mathbb{R}$. Next, we show that

$$\Omega(\gamma') \subseteq \{\mathcal{D} \in (S\times\mathbb{R}_{\geq 0})^{n^2} \mid \text{for } i \in \{1,\dots,n\}, \quad (16)$$
$$\|p_i^i - C_{\text{gV}_i}(\pi_{\mathcal{A}^i}(\mathcal{D}^i))\| \leq \text{bnd}(\pi_{\mathcal{A}^i}(\mathcal{D}^i))\}.$$

We reason by contradiction. Assume there exists $\mathcal{D} \in \Omega(\gamma)$ for which there is $i \in \{1,\dots,n\}$ such that $\|p_i^i - C_{\text{gV}_i}(\pi_{\mathcal{A}^i}(\mathcal{D}^i))\| > \text{bnd}(\pi_{\mathcal{A}^i}(\mathcal{D}^i))$. Then, using Lemmas 2.1 and 5.1 together with Proposition 5.2, we deduce that any possible evolution from $\mathcal{D}$ under T_{sync} will strictly decrease $\mathcal{H}$, which is a contradiction with the fact that $\Omega(\gamma')$ is weakly positively invariant for T_{sync}.

Furthermore, note that for each i, the inequality $\text{bnd}_i < \max\{\|p_i^i - C_{\text{gV}_i}\|, \varepsilon\}$ is satisfied at $\mathcal{D}'(t_\ell)$, for all $\ell \in \mathbb{Z}_{\geq 0}$. Therefore, by continuity, it also holds on $\Omega(\gamma')$, i.e.,

$$\text{bnd}(\pi_{\mathcal{A}^i}(\mathcal{D}^i)) \leq \max\{\|p_i^i - C_{\text{gV}_i}(\pi_{\mathcal{A}^i}(\mathcal{D}^i))\|, \varepsilon\}, \quad (17)$$

for all $i \in \{1,\dots,n\}$ and all $\mathcal{D} \in \Omega(\gamma')$. Let us now show that $\Omega(\gamma') \subseteq \{\mathcal{D} \in (S\times\mathbb{R}_{\geq 0})^{n^2} \mid \text{for } i \in \{1,\dots,n\}, p_i^i = C_{V_i}\}$. Consider $\widetilde{\mathcal{D}} \in \Omega(\gamma')$. Since $\Omega(\gamma')$ is weakly positively invariant, there exists $\widetilde{\mathcal{D}}_1 \in \Omega(\gamma') \cap T_{\text{sync}}(\widetilde{\mathcal{D}})$. Note that (16) implies that $\text{loc}(\widetilde{\mathcal{D}}_1) = \text{loc}(\widetilde{\mathcal{D}})$. We consider two cases, depending on whether or not agents have got up-to-date information in $\widetilde{\mathcal{D}}_1$. If agent i gets up-to-date information, then $\text{bnd}(\pi_{\mathcal{A}^i}(\widetilde{\mathcal{D}}_1^i)) = 0$, and consequently, from (16), $p_i^i = p_i'^i = C_{\text{gV}_i}(\pi_{\mathcal{A}^i}(\widetilde{\mathcal{D}}_1^i) = C_{V_i}$, and the result follows. If agent i does not get up-to-date information, then $\text{bnd}(\pi_{\mathcal{A}^i}(\widetilde{\mathcal{D}}_1^i)) > \text{bnd}(\pi_{\mathcal{A}^i}(\widetilde{\mathcal{D}}^i))$ and $\text{gV}_i(\pi_{\mathcal{A}^i}(\widetilde{\mathcal{D}}_1^i)) \subset \text{gV}_i(\pi_{\mathcal{A}^i}(\widetilde{\mathcal{D}}^i))$ by Lemma 4.1. Again, using the fact that $\Omega(\gamma')$ is weakly positively invariant set, there exists $\widetilde{\mathcal{D}}_2 \in \Omega(\gamma') \cap T_{\text{sync}}(\widetilde{\mathcal{D}}_1)$. Reasoning repeatedly in this way, the only case we need to discard is when agent i never gets up-to-date information. In such a case, $\|p_i^i - C_{\text{gV}_i}\| \to 0$ while bnd_i monotonically increases towards $\text{diam}(S)$. For sufficiently large ℓ, we have that $\|p_i^i - C_{\text{gV}_i}(\pi_{\mathcal{A}^i}(\widetilde{\mathcal{D}}_\ell^i))\| < \varepsilon$. Then (17) implies $\text{bnd}_i(\pi_{\mathcal{A}^i}(\widetilde{\mathcal{D}}_\ell^i)) < \varepsilon$, which contradicts the fact that $\text{bnd}(\pi_{\mathcal{A}^i}(\widetilde{\mathcal{D}}_\ell^i))$ tends to $\text{diam}(S)$. This ends the proof. □

Remark 6.4 (Convergence with errors in position information) A convergence result similar to Proposition 6.1 can be stated in the case when errors in position information are present, as discussed in Remark 3.1. In this case, for sufficiently small maximum position error δ, it can be shown (although we do not do it here for reasons of space) that the network will converge to within a constant factor of δ of the set of centroidal Voronoi configurations. •

7 Extensions

In this section, we briefly discuss two important variations of the `self-triggered centroid algorithm`. Section 7.1 discusses a procedure that agents can implement to decrease their maximum velocity as they get close to their optimal locations. Section 7.2 discusses the convergence of asynchronous executions.

7.1 Maximum velocity decrease

The agents update their individual memories along the execution of the `self-triggered centroid algorithm` by growing the regions of uncertainty about the position of other agents at a rate $v_{\max}$. However, as the network gets close to the optimal configuration (as guaranteed by Proposition 6.1), agents move at velocities much smaller than the nominal maximum velocity $v_{\max}$ per timestep. Here, we describe a procedure that the network can implement to diminish this mismatch and reduce the need for up-to-date location information.

The strategy is based on the simple observation that the gradient $\nabla\mathcal{H}$ of the objective function vanishes exactly on the set of centroidal Voronoi configurations. Therefore, as the network gets close to this set, the norm of $\nabla\mathcal{H}$ tends to zero. From [Du et al., 1999, Bullo et al., 2009], we know that $\frac{\partial\mathcal{H}}{\partial p_i} = 2M_{V_i}(p_i - C_{V_i})$ for each $i \in \{1,\dots,n\}$, and hence

$$\left\|\frac{\partial\mathcal{H}}{\partial p_i}\right\| \leq 2M_{\text{dgV}_i}(\|p_i - C_{\text{gV}_i}\| + \text{bnd}_i).$$

Note that this upper bound is computable by agent i. The objective of the network is then to determine if, for a given design parameter δ, with $0 < \delta \ll 1$,

$$2M_{\text{dgV}_i}(\|p_i - C_{\text{gV}_i}\| + \text{bnd}_i) < \delta \quad (18)$$

for all $i \in \{1,\dots,n\}$. This check can be implemented in a number of ways. Here, we use a convergecast algorithm, see e.g., [Peleg, 2000].

The strategy can informally be described as follows. Each time an agent i communicates with its neighbors, it checks if (18) is satisfied for $\mathcal{A}^i \cup \{i\}$. If this is the case, then agent i triggers the computation of a spanning tree (e.g., a breadth-first-search spanning tree [Peleg, 2000])

rooted at itself which is used to broadcast the message 'the check is running'. An agent j passes this message to its children or sends an acknowledgement to its parent if and only if (18) is satisfied for j. At the end of this procedure, the root i has the necessary information to determine if (18) holds for all agents. If this is the case, agent i broadcasts a message to all agents to set $v_{\max}^{+} = v_{\max}/2$ and $\delta^{+} = \delta/2$.

7.2 Asynchronous executions

Here, we relax assumption (i) in Section 3 and consider asynchronous executions of the `self-triggered centroid algorithm`. We begin by describing a totally asynchronous model for the operation of the network agents, cf. [Bertsekas and Tsitsiklis, 1997]. Let $\mathcal{T}^i = \{t_0^i, t_1^i, t_2^i, \dots\} \subset \mathbb{R}_{\geq 0}$ be a time schedule for agent $i \in \{1, \dots, n\}$. Assume agent i executes the algorithm according to $\mathcal{T}^i$, i.e., the agent executes the steps `1:-12:` described in Table 5 at time t_ℓ^i, for $\ell \in \mathbb{Z}_{\geq 0}$, with timestep $(t_{\ell+1}^i - t_\ell^i)$ instead of Δt. In general, the time schedules of different agents do not coincide and this results in an overall asynchronous execution. Our objective is to show that, under mild conditions on the time schedules of the agents, one can establish the same asymptotic convergence properties for asynchronous evolutions.

Our analysis strategy has two steps. First, we synchronize the network operation using the procedure of analytic synchronization, see e.g., [Lin et al., 2007]. Second, we use this to lay out a proof strategy similar to the one used for the synchronous case.

7.2.1 Analytic synchronization

Analytic synchronization is a procedure that consists of merging together the individual time schedules $\mathcal{T}^i$, $i \in \{1, \dots, n\}$, of the network agents into a global time schedule $\mathcal{T} = \{t_0, t_1, t_2, \dots\}$ by setting

$$\mathcal{T} = \cup_{i=1}^{n} \mathcal{T}^i.$$

This synchronization is performed only for analysis purposes, i.e., $\mathcal{T}$ is unknown to the individual agents. Note that more than one agent may be active at any given $t \in \mathcal{T}$. For convenience, we define

$$\Delta t_\ell = t_{\ell+1} - t_\ell > 0,$$

for $\ell \in \mathbb{Z}_{\geq 0}$, i.e., Δt_ℓ is the time from t_ℓ until at least one agent is active again.

7.2.2 Convergence of asynchronous executions

The procedure of analytic synchronization allows us to analyze the convergence properties of asynchronous executions mimicking the proof strategy used in Section 6 for the synchronous case. We do not include the full proof here to avoid repetition. Instead, we provide the necessary elements to carry it over.

The main tool is the definition of a set-valued map T_{async} whose trajectories include the asynchronous executions of the `self-triggered centroid algorithm`. As before, the construction of T_{async} is divided in two parts, a first one that captures the agents' motion and uncertainty update to the network memory, and a second one that captures the acquisition of up-to-date information. The definition of T_{async} also takes into account the global time schedule $\mathcal{T}$ in order to capture the different schedules of the agents. For convenience, we define the network state to be $(x, \ell) \in ((S \times \mathbb{R}_{\geq 0})^n \times S)^n \times \mathbb{Z}_{\geq 0}$, where

$$x = ((\mathcal{D}^1, u^1), \dots, (\mathcal{D}^n, u^n)),$$

u^i denotes the waypoint of agent $i \in \{1, \dots, n\}$ and ℓ is a time counter. For ease of notation, let

$$(S \times \mathbb{R}_{\geq 0})_e^{n^2} = ((S \times \mathbb{R}_{\geq 0})^n \times S)^n \times \mathbb{Z}_{\geq 0}.$$

Motion and uncertainty update. The motion and time update map $\mathcal{M} : (S \times \mathbb{R}_{\geq 0})_e^{n^2} \to (S \times \mathbb{R}_{\geq 0})_e^{n^2}$ simply corresponds to all agents moving towards their waypoints while increasing in their memories the uncertainty about the locations of other agents. The map is given by $\mathcal{M}(x, \ell) = (\mathcal{M}_1(x, \ell), \dots, \mathcal{M}_n(x, \ell), \ell)$ where

$$\begin{aligned}\mathcal{M}_i(x, \ell) = \big(&(p_1^i, \min\{r_1^i + v_{\max}\Delta t_\ell, \operatorname{diam}(S)\}), \dots, \\ &\big(\operatorname{tbb}(p_i^i, v_{\max}\Delta t_\ell, u^i, 0), 0\big), \dots, \\ &(p_n^i, \min\{r_n^i + v_{\max}\Delta t_\ell, \operatorname{diam}(S)\}, u^i)\big).\end{aligned}$$

Note that $\operatorname{tbb}(p_i^i, v_{\max}\Delta t_\ell, u^i, 0)$ corresponds to where agent i can get to in time Δt_ℓ while moving towards its waypoint u^i. The map $\mathcal{M}$ is continuous.

Acquisition of up-to-date information. Any given time might belong to the time schedules of only a few agents. Moreover, these agents are faced with the decision of whether to acquire up-to-date information about the location of other agents. This is captured with the set-valued map $\mathcal{U} : (S \times \mathbb{R}_{\geq 0})_e^{n^2} \rightrightarrows (S \times \mathbb{R}_{\geq 0})_e^{n^2}$. Given the global time schedule $\mathcal{T}$, the map $\mathcal{U}$ associates the Cartesian product $\mathcal{U}(x, \ell)$ whose $(n+1)$th component is $\ell + 1$ and whose ith component, $i \in \{1, \dots, n\}$, is one of the following three possibilities: either (i) the vector

$$(\mathcal{D}^i, u^i),$$

which means i is not active at time step ℓ, (ii) the vector

$$(\mathcal{D}^i, \operatorname{tbb}(p_i^i, v_{\max}(t_{\ell'} - t_\ell), C_{\mathrm{gV}_i}, \operatorname{bnd}_i)),$$

for some $\ell' > \ell$, with $C_{\mathrm{gV}_i} = C_{\mathrm{gV}_i}(\pi_{\mathcal{A}^i}(\mathcal{D}^i))$, $\operatorname{bnd}_i = \operatorname{bnd}(\pi_{\mathcal{A}^i}(\mathcal{D}^i))$, and $\mathcal{A}^i = \{i\} \cup \operatorname{argmin}_{j \in \{1,\dots,n\} \setminus \{i\}} r_j^i$, which means i is active at time step ℓ and recomputes its waypoint but does not get any up-to-date information, or (iii) the vector

$$((p_1', r_1'), \dots, (p_n', r_n'), \operatorname{tbb}(p_i^i, v_{\max}(t_{\ell'} - t_\ell), C_{\mathrm{gV}_i}, \operatorname{bnd}_i)),$$

for some $\ell' > \ell$, where $(p'_j, r'_j) = (p^j_j, 0)$ for $j \in \{i\} \cup \mathcal{N}_i$ and $(p'_j, r'_j) = (p^j_j, r^j_j)$ otherwise, which means i is active at time step ℓ, gets up-to-date information and recomputes its waypoint. In this case, $C_{\mathrm{gV}_i} = C_{\mathrm{gV}_i}(\pi_{\mathcal{A}^i}((p'_1, r'_1), \ldots, (p'_n, r'_n)))$, $\mathrm{bnd}_i = \mathrm{bnd}(\pi_{\mathcal{A}^i}((p'_1, r'_1), \ldots, (p'_n, r'_n)))$, and $\mathcal{A}^i = \{i\} \cup \mathcal{N}_i$. The set-valued map $\mathcal{U}$ is closed.

Finally, we define the set-valued map $T_{\mathrm{async}} : (S \times \mathbb{R}_{\geq 0})^{n^2}_e \rightrightarrows (S \times \mathbb{R}_{\geq 0})^{n^2}_e$ by $T_{\mathrm{async}} = \mathcal{U} \circ \mathcal{M}$. Given the continuity of $\mathcal{M}$ and the closedness of $\mathcal{U}$, the map T_{async} is closed. Moreover, the asynchronous executions of the `self-triggered centroid algorithm` with time schedules $\mathcal{T}^i$, $i \in \{1, \ldots, n\}$ are trajectories of

$$(x(\ell + 1), \ell + 1) \in T_{\mathrm{async}}(x(\ell), \ell).$$

Equipped with the definition of T_{async}, one can now reproduce the proof strategy followed in Section 6 and establish the monotonic evolution of the objective function, the weakly positively invariant nature of the omega limit sets of its trajectories, and finally, the same asymptotic convergence properties of the asynchronous executions of the `self-triggered centroid algorithm`, which we state here for completeness.

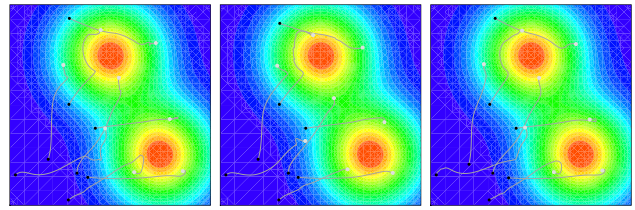
Proposition 7.1 *Assume the time schedules $\mathcal{T}^i$, $i \in \{1, \ldots, n\}$ are infinite and unbounded. For $\varepsilon \in [0, \mathrm{diam}(S))$, the agents' position evolving under the asynchronous* `self-triggered centroid algorithm` *with time schedules $\mathcal{T}^i$, $i \in \{1, \ldots, n\}$ from any initial network configuration in S^n converges to the set of centroidal Voronoi configurations.*

8 Simulations

Here, we provide several simulations to illustrate our results. All simulations are done with $n = 8$ agents moving in a 4m $\times$ 4m square, with a maximum velocity $v_{\max} = 1$m/s. The synchronous executions operate with $\Delta t = .025$s. In the asynchronous execution shown in Figure 4(b), agents in $\{1, 2, 3, 4\}$ and $\{5, 6, 7, 8\}$ share their time schedules, respectively. These time schedules are generated as follows: the four first time steps are randomly generated, and then they repeat periodically. We compare our algorithm against the move-to-centroid strategy where agents have perfect location information at all times, see [Cortés et al., 2004]; we refer to this as the benchmark case. For each agent $i \in \{1, \ldots, n\}$, we adopt the following model [Firouzabadi, 2007] for quantifying the total power $\mathcal{P}_i$ used by agent i to communicate, in $dBmW$ power units,

$$\mathcal{P}_i = 10 \log_{10} \Big[\sum_{j \in \{1, \ldots, n\}, i \neq j}^{n} \beta 10^{0.1 P_{i \to j} + \alpha \|p_i - p_j\|} \Big],$$

where $\alpha > 0$ and $\beta > 0$ depend on the characteristics of the wireless medium and $P_{i \to j}$ is the power received by j of the signal transmitted by i in units of $dBmW$. In our simulations, all these values are set to 1.



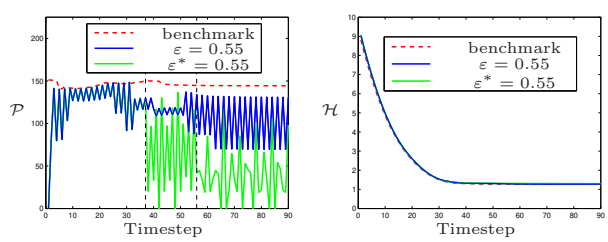
(a) (b) (c)

Fig. 4. Network trajectories of (a) a synchronous execution and (b) an asynchronous execution of the `self-triggered centroid algorithm` with $\varepsilon = 0.25$, and (c) the centroid algorithm with perfect information at all times (benchmark case). The black and grey dots correspond to the initial and final agent positions, respectively.

Figures 4 and 5 illustrate an execution of the `self-triggered centroid algorithm` for a density ϕ which is a sum of two Gaussian functions

$$\phi(x) = e^{-\|x - q_1\|^2} + e^{-\|x - q_2\|^2},$$

with $q_1 = (2, 3)$ and $q_2 = (3, 1)$, and compare its performance against the benchmark case. The communication power in a given timestep is the sum of the energy required for all the directed point-to-point messages to be sent in that timestep. Additionally, Figure 5 shows an execution that is also incorporating the distributed algorithm for decreasing velocity.



(a) (b)

Fig. 5. Communication power $\mathcal{P}$ used by the network (a) and value of $\mathcal{H}$ (b) at each time step of the executions in Figure 4(a) and (c), and an execution with the maximum velocity decrease (denoted with ε^*). The vertical lines denote the timesteps where agents reduce their maximum velocity.

Figure 6 shows the average communication power expenditure and the average time to convergence of the `self-triggered centroid algorithm` for varying ε over 20 random initial agent positions based on uniformly sampling the domain. One can see how as ε gets larger, the communication effort of the agents decreases at the cost of a slower convergence on the value of $\mathcal{H}$. Interestingly, for small ε, the network performance does not deteriorate significantly while the communication effort by the individual agents is substantially smaller. The lower cost associated to the `self-triggered`

`centroid algorithm` is due to requiring less communication than the benchmark case.

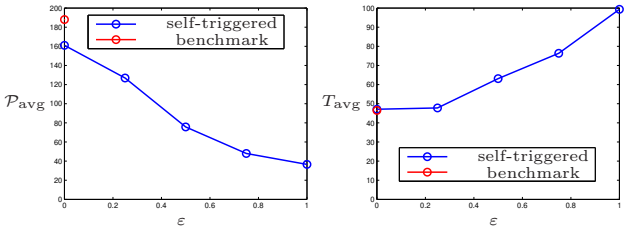


Fig. 6. Plots of the average (a) communication power consumption $\mathcal{P}_{\text{avg}}$ and (b) timesteps to convergence T_{avg} over 20 simulations for varying ε.

9 Conclusions

We have proposed the `self-triggered centroid algorithm`. This strategy combines an update law to determine when old information needs to be refreshed and a motion control law that uses this information to decide how to best move. We have analyzed the correctness of both synchronous and asynchronous executions of the proposed algorithm using tools from computational geometry and set-valued analysis. Our results have established the same convergence properties that a synchronous algorithm with perfect information at all times would have. Extensive simulations have illustrated the substantial communication savings of the `self-triggered centroid algorithm`, which can be further improved by employing an event-triggered strategy to prescribe maximum velocity decreases as the network gets closer to its final configuration. In future work, we plan to characterize analytically the tradeoff between performance and communication cost, provide guarantees on the network energy savings, and explore the extension of these ideas to other coordination tasks.

References

A. Anta and P. Tabuada. To sample or not to sample: self-triggered control for nonlinear systems. *IEEE Transactions on Automatic Control*, 55(9):2030–2042, 2010.

D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997. ISBN 1886529019.

F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press, 2009. ISBN 978-0-691-14195-4. Electronically available at http://coordinationbook.info.

C. G. Cassandras and S. Lafortune. *Introduction to Discrete-Event Systems*. Springer, 2 edition, 2007. ISBN 0387333320.

H. Choset. Coverage for robotics – A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):113–126, 2001.

J. Cortés, S. Martínez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2):243–255, 2004.

J. Cortés, S. Martínez, and F. Bullo. Spatially-distributed coverage optimization and control with limited-range interactions. *ESAIM. Control, Optimisation & Calculus of Variations*, 11 (4):691–719, 2005.

D. V. Dimarogonas and K. H. Johansson. Event-triggered control for multi-agent systems. In *IEEE Conf. on Decision and Control*, pages 7131–7136, Shanghai, China, 2009.

Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4): 637–676, 1999.

S. Firouzabadi. Jointly optimal placement and power allocation in wireless networks. Master's thesis, University of Maryland at College Park, 2007.

A. Howard, M. J. Matarić, and G. S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed scalable solution to the area coverage problem. In *Int. Conference on Distributed Autonomous Robotic Systems*, pages 299–308, Fukuoka, Japan, June 2002.

I. I. Hussein and D. M. Stipanovìc. Effective coverage control for mobile sensor networks with guaranteed collision avoidance. *IEEE Transactions on Control Systems Technology*, 15(4):642–657, 2007.

M. Jooyandeh, A. Mohades, and M. Mirzakhah. Uncertain Voronoi diagram. *Information Processing Letters*, 109(13): 709–712, 2009.

K. Kang, J. Yan, and R. R. Bitmead. Cross-estimator design for coordinated systems: Constraints, covariance, and communications resource assignment. *Automatica*, 44(5):1394–1401, 2008.

A. Kwok and S. Martínez. Deployment algorithms for a power-constrained mobile sensor network. *International Journal on Robust and Nonlinear Control*, 20(7):725–842, 2010.

J. Lin, A. S. Morse, and B. D. O. Anderson. The multi-agent rendezvous problem. Part 2: The asynchronous case. *SIAM Journal on Control and Optimization*, 46(6):2120–2147, 2007.

M. Mazo Jr. and P. Tabuada. Decentralized event-triggered control over wireless sensor/actuator networks. *IEEE Transactions on Automatic Control*, 2011. To appear.

A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. Wiley Series in Probability and Statistics. Wiley, 2 edition, 2000. ISBN 0471986356.

M. Pavone, A. Arsie, E. Frazzoli, and F. Bullo. Distributed algorithms for environment partitioning in mobile robotic networks. *IEEE Transactions on Automatic Control*, 56(9), 2011. To appear.

D. Peleg. *Distributed Computing. A Locality-Sensitive Approach*. Monographs on Discrete Mathematics and Applications. SIAM, 2000. ISBN 0898714648.

M. Schwager, D. Rus, and J. J. Slotine. Decentralized, adaptive coverage control for networked robots. *International Journal of Robotics Research*, 28(3):357–375, 2009.

J. Sember and W. Evans. Guaranteed Voronoi diagrams of uncertain sites. In *Canadian Conference on Computational Geometry*, Montreal, Canada, 2008.

R. Subramanian and F. Fekri. Sleep scheduling and lifetime maximization in sensor networks. In *Symposium on Information Processing of Sensor Networks*, pages 218–225, New York, NY, 2006.

M. Velasco, P. Marti, and J. M. Fuertes. The self triggered task model for real-time control systems. In *Proceedings of the 24th IEEE Real-Time Systems Symposium*, pages 67–70, 2003.

P. Wan and M. D. Lemmon. Event-triggered distributed optimization in sensor networks. In *Symposium on Information Processing of Sensor Networks*, pages 49–60, San Francisco, CA, 2009.

X. Wang and M. D. Lemmon. Self-triggered feedback control systems with finite-gain L2 stability. *IEEE Transactions on Automatic Control*, 54(3):452–467, 2009.