Self-triggered coordination of robotic networks for optimal deployment

C. Nowzari, J. Cortés
Department of Mechanical and Aerospace Engineering, University of California, San Diego, CA, 92093, USA

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, Martínez, Karatas, & Bullo, 2004), where distributed algorithms based on centroidal Voronoi partitions are presented, and (Cortés, Martínez, & Bullo, 2005), where limited-range interactions are considered. Other works on deployment coverage problems include (Howard, Matarić, & Sukhatme, 2002; Kwok & Martínez, 2010; Pavone, Arsie, Frazzoli, & Bullo, 2011; Schwager, Rus, & Slotine, 2009). 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 & Stipanović, 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 & Lafortune, 2007), and the research in triggered control (Anta & Tabuada, 2010; Subramanian & Fekri, 2006; Velasco, Marti, & Fuertes, 2003; Wang & Lemmon, 2009), 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, Yan, & Bitmead, 2008) on collision avoidance while performing point-to-point reconfiguration, (Dimarogonas & Johansson, 2009) on achieving agreement, (Wan & Lemmon, 2009) on distributed optimization, and (Mazo & 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 information, 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 \([p, q] \subseteq \mathbb{R}^d\) be the closed segment with extreme points \(p\) and \(q \in \mathbb{R}^d\), let \( B(p, r) = \{ q \in \mathbb{R}^d \mid \| q - p \| \leq r \} \) and \( H_p = \{ q \in \mathbb{R}^d \mid \| q - p \| \leq \| q - o \| \} \) be the closed halfspace determined by \( p, o \in \mathbb{R}^d \) that contains \( p \). Let \( \phi : \mathbb{R}^d \rightarrow \mathbb{R}_{\geq 0} \) be a bounded measurable function, termed density. For \( S \subseteq \mathbb{R}^d \), the mass and center of mass of \( S \) with respect to \( \phi \) are

\[
M_S = \int_S \phi(q) dq, \quad C_S = \frac{1}{M_S} \int_S \phi(q) dq.
\]

The circumsphere \( cc_S \) of a bounded set \( S \subseteq \mathbb{R}^d \) is the center of the closed ball of minimum radius that contains \( S \). The circumsphere \( cc_S \) of \( S \) is the radius of this ball. The diameter of \( S \) is \( \text{diam}(S) = \max_{p \in S} \| p - q \| \).

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

\[
\begin{cases}
\{ p + \delta \text{unit}(q-p) \} & \text{if } \| p - pr_{B(q,r)}(p) \| > \delta, \\
pr_{B(q,r)}(p) & \text{if } \| p - pr_{B(q,r)}(p) \| \leq \delta.
\end{cases}
\]

Fig. 1 illustrates the action of \( \text{tbb} \).

The notion of Voronoi partitioning (Okabe, Boots, Sugihara, & Chiu, 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 \( V_i = \{ q \in S \mid \| q - p_i \| \leq \| q - p_j \|, \forall j \neq i \} \).

2.2. Facility location and aggregate distortion

We introduce here a locational optimization function termed aggregate distortion (Bullo, Cortés, & Martínez, 2009; Du, Faber, & Gunzburger, 1999). Consider a set of agent positions \( P \subseteq \mathbb{R}^n \). The agent performance at \( q \) of the agent \( p_i \) degrades with \( \| q - p_i \|^2 \).

Assume a density \( \phi : S \rightarrow \mathbb{R} \) is available, with \( \phi(q) \) reflecting the likelihood of an event happening at \( q \). Consider then the minimization of

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

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

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

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

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

where \( W \) is a partition of \( S \), and the \( i \)th 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 \( W \). The next result (Bullo et al., 2009; Du et al., 1999) characterizes its critical points.

Lemma 2.1. Given \( P \in S^n \) and a partition \( W \) of \( S \),

\[
\mathcal{H}(P, \mathcal{V}(P)) \leq \mathcal{H}(P, W),
\]

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

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

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 $t_i = t_0 + i \Delta t$, for $i \in \mathbb{Z}_{\geq 0}$;

(ii) each agent can move at a maximum speed of $v_{\text{max}}$, i.e., $\|p_i(t_i + \Delta t) - p_i(t_i)\| \leq v_{\text{max}} \Delta t$;

(iii) for $p_{\text{goal}} \in S$, there exists a control such that $\|p_i(t_i + \Delta t) - p_{\text{goal}}\| < \|p_i(t_i) - p_{\text{goal}}\|$, $p_i(t_i + \Delta t) \in [p_i(t_i), p_{\text{goal}}]$ and $p_i([t_i, t_{i+1}]) \subset S$.

Later in Section 7.2, we will relax assumption (i). In our later developments, we assume in (iii) that, if $\|p_i(t_i) - p_{\text{goal}}\| \leq v_{\text{max}} \Delta t$, then $p_i(t_i + \Delta t) = p_{\text{goal}}$ 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 the energy that agents expend to communicate scales with distance, agents have to balance the need for updated information with the desire of spending 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 $p_j^i$ and the time elapsed $r_j^i \in \mathbb{R}_{\geq 0}$ since this information was received (if $i$ does not ever receive information about $j$, then $p_j^i$ and $r_j^i$ are never initiated). For itself, agent $i$ has access to up-to-date location information, i.e., $p_j^i = p_j$ and $r_j^i = 0$ at all times. When data is available, agent $i$ knows that, at the current time, $t$, it will not have traveled more than $r_j^i = \max_{j' \neq i} r_j^{j'}$ from $p_j^i$, and hence it can construct a ball $B(p_j^i, r_j^i)$ that is guaranteed to contain the true location of $j$. Note that once any radius $r_j^i$ becomes diam$(S)$, it does not make sense to grow it any more. The data is stored in

$$\mathcal{D} = ((p_1^1, r_1^1), \ldots, (p_n^n, r_n^n)) \in (S \times \mathbb{R}_{\geq 0})^n. \tag{4}$$

Additionally, agent $i$ maintains a set $A^i \subset \{1, \ldots, n\}$ with $i \in A^i$ that, at any time $t$, corresponds to the agents whose position information should be used. For instance, $A^i = \{1, \ldots, n\}$ would mean that agent $i$ uses all the information contained in $\mathcal{D}^i$, although this is not necessary as we will explain in Section 5.2. In fact, an individual agent does not need to know the total number of agents in the network or maintain a place holder for every other agent in its memory. We have only chosen to define the agent memory as in (4) to simplify the exposition of the technical arguments later.

We refer to $\mathcal{D} = (\mathcal{D}_1, \ldots, \mathcal{D}_n) \in (S \times \mathbb{R}_{\geq 0})^n$ as the entire memory of the network. We find it convenient to define the map $\text{loc} : (S \times \mathbb{R}_{\geq 0})^n \rightarrow S^n, \text{loc}(\mathcal{D}) = (p_1^1, \ldots, p_n^n)$, 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 be easily incorporated into the model if they are upper bounded by $\delta \in \mathbb{R}_{\geq 0}$ by setting $r_j^{i'} = \max_{j \in A^i} r_j^i + \delta$, 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 [Jooyandeh, Mohades, & Mirzakhah, 2009; Sember & Evans, 2008]. 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 $gV(D_1, \ldots, D_n) = \{gV_1, \ldots, gV_n\}$.

$$gV_i = \{ q \in S \mid \max_{x \in D_i} \|q - x\| = \min_{x \neq y} \|q - y\| \text{ for all } j \neq i \}. \tag{5}$$

With a slight abuse of notation, $gV_i(D)$ denotes the $i$th component of $gV(D_1, \ldots, D_n)$. Note that $gV_i$ contains the points of $S$ that are guaranteed to be closer to $p_i$ than to any other node $p_j, j \neq i$. Due to the uncertainties in positions, 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 Fig. 2(a). Each point in the boundary of $gV_i$ belongs to the set

$$\Delta^i_S = \{ q \in S \mid \max_{x \in D_i} \|q - x\| = \min_{x \neq y} \|q - y\| \}. \tag{6}$$

for some $j \neq i$. Note that $\Delta^i_S \neq \Delta^i_p$. If every region $D_i$ is a point, $D_i = \{p_i\}$, then $gV(D_1, \ldots, D_n) = \mathcal{V}(p_1, \ldots, p_n)$. 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., $gV_i \subset \mathcal{V}_i, i \in \{1, \ldots, n\}$. Agent $p_i$ is a guaranteed Voronoi neighbor of $p_j$ if $\Delta^i_S \cap \partial gV_i$ is not empty nor a singleton. The set of guaranteed Voronoi neighbors of agent $i$ is $g\mathcal{N}_i(D)$.

Throughout the paper, we consider uncertain regions given by balls, $D_i = B(p_i, r_i), i \in \{1, \ldots, n\}$. Then, the edges (6) composing the boundary of $gV_i$ are given by

$$\Delta^i_S = \{ q \in S \mid \|q - p_i\| + r_i = \|q - p_j\| - r_j \}. \tag{7}$$

thus they lie on the arm of the hyperbola closest to $p_i$ with foci $p_j$ and $p_i$, 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 $D_i = B(p_i, r_i)$ and $D'_i = B(p_i + r_i, a)$, 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 $j \in g\mathcal{N}_i(D'_i)$. This fact implies, according to (6), that there exists $q \in S$ such that

$$\|q - p_i\| + r_i + a = \|q - p_j\| - r_j - a < \|q - p_j\| - r_j,$$

for all $k \in \{1, \ldots, n\} \setminus \{i, j\}$ and $q \neq p_i$. 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 $\|q' - p_i\| = \|q - p_i\| - \|q' - q\|$. Therefore, we can write

$$\|q' - p_i\| - r_j = \|q' - p_i\| - r_j - \|q - q'\|$$

$$< \|q - p_i\| - r_j - \|q - q'\|,$$

for all $k \in \{1, \ldots, n\} \setminus \{i, j\}$, where we have used (7). Now, using the triangle inequality $\|q - p_k\| \leq \|q - q'\| + \|q' - p_k\|$, we deduce $\|q - p_k\| - r_k < \|q - p_k\| - r_k$, for all $k \in \{1, \ldots, n\} \setminus \{i, j\}$, and hence $j \in g\mathcal{N}_i(D'_i)$. □
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 
\[
dg \mathcal{V}(D_1, \ldots, D_n) = \{ \dg V_1, \ldots, \dg V_n \}
\]
defined by 
\[
dg V_i = \left\{ q \in S \mid \min_{x \in D_i} \| q - x \| \leq \max_{y \in D_j} \| q - y \| \text{ for all } j \neq i \right\}.
\]
With a slight abuse of notation, \( \dg V_i \) denotes the \( i \)-th component of \( \dg \mathcal{V}(D_1, \ldots, D_n) \). Note that the points of \( S \) outside \( \dg V_i \) 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 Fig. 2(b). Each point in the boundary of \( \dg V_i \) belongs to the set 
\[
\Delta_g^{dg} = \left\{ q \in S \mid \min_{x \in D_i} \| q - x \| = \max_{y \in D_j} \| q - y \| \right\},
\]
for some \( j \neq i \). Note that \( \Delta_g^{dg} \neq \Delta_g^{dg} \). If every region \( D_i \) is a point, \( D_i = \{ p_i \} \), then \( \dg V_i(D_1, \ldots, D_n) = \mathcal{V}(p_1, \ldots, p_n) \). For any collection of points \( D_i \), \( i \in \{ 1, \ldots, n \} \), the guaranteed Voronoi covering contains the Voronoi partition, i.e., \( V \subset \dg V_i \), \( i \in \{ 1, \ldots, n \} \). Agent \( p_i \) is a dual guaranteed Voronoi neighbor of \( p_i \) if \( \Delta_g^{dg} \cap \dg V_i \) is not empty nor a singleton. The set of dual guaranteed Voronoi neighbors of \( i \) is \( \dg V_i \).

Consider the uncertain regions given \( D_i = B(p_i, r_i) \), \( i \in \{ 1, \ldots, n \} \). The edges (8) are given by 
\[
\Delta_g^{dg} = \{ q \in S \mid \| q - p_i \| - r_i = \| q - p_j \| + r_j \},
\]
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 \( r_1, \ldots, r_n \), \( a \in \mathbb{R}_{>0} \), let \( D_i = B(p_i, r_i) \) and \( D'_i = B(p_i, r_i + a) \), for \( i \in \{ 1, \ldots, n \} \). Then, 
\( \dg V_i(D_1, \ldots, D_n) \subset \dg V_i(D'_1, \ldots, D'_n) \), for all \( i \in \{ 1, \ldots, n \} \).

The next result follows from the definition of \( \dg V_i \).

**Lemma 4.3.** For \( D_1, \ldots, D_{n+1} \subset S \) and all \( i \in \{ 1, \ldots, n \} \), 
\( \dg V_i(D_1, \ldots, D_n, D_{n+1}, \ldots, D_{n+1}) \subseteq \dg V_i(D_1, \ldots, D_n) \).

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 \( \phi \) 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 \( \angle (p - q^* - q' - p') = \pi - \angle (p - p', q^* - p') \).

Now,
\[
(p - p') \cdot (q^* - p') = (p - q^* + q^* - p') \cdot (q^* - p') = (p - q^*) \cdot (q^* - p') + \| q^* - p' \|^2.
\]
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 
\[
\| q^* - q \|^2 = \| q^* - p' \|^2 + \| q - p' \|^2 - 2 \| q^* - p' \| \| q - p' \| \cos \angle (q - p', q^* - p') > \| q - p' \|^2,
\]
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 (10) contradicts the hypothesis \( \| p' - q \| \geq \| q^* - q \| \). □
Hence, with the notation of Lemma 5.1, if \( p = p_i \) computes the goal \( q = C_{gVi} \) and moves towards it to \( p' \), then the distance to \( q^* = C_{Vi} \) decreases as long as

\[
\|p' - C_{gVi}\| \geq \|C_{Vi} - C_{gVi}\| \quad (11)
\]

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

**Proposition 5.2.** Let \( L \subset V \subset U \). Then, for any density function \( \phi \), the following holds

\[
\|C_{Vi} - C_{L}\| \leq 2c_r a \left( 1 - \frac{M_i}{M_U} \right) . \quad (12)
\]

**Proof.** For convenience, let \( a = M_U, b = M_V, \) and \( c = M_i \). By hypothesis, \( a \leq b \leq c \). Note also that \( M_U/L = c - a \). By definition, we have

\[
C_{Vi} - C_{L} = \frac{1}{b} \int_V q \phi(q)dq - \frac{1}{a} \int_V q \phi(q)dq.
\]

(13)

For any \( v \in \mathbb{R}^2, v = \frac{1}{b} \int_V v \phi(q)dq = \frac{1}{a} \int_V v \phi(q)dq. \) Summing and subtracting \( v \in \mathbb{R}^2 \), we get that (13) equals

\[
\frac{1}{b} \int_V (q - v) \phi(q)dq - \frac{1}{a} \int_V (q - v) \phi(q)dq
\]

\[
= \frac{1}{b} \int_V (q - v) \phi(q)dq + \left( \frac{1}{b} - \frac{1}{a} \right) \int_V (q - v) \phi(q)dq.
\]

Taking norms, we deduce that for \( v = ccu \), we have

\[
\|C_{Vi} - C_{L}\| \leq \frac{1}{b} c_r U(b - a) + \frac{1}{a} - \frac{1}{b} c_r U a.
\]

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 (12) is an equality. Exploiting Proposition 5.2, agent \( i \) can use \( L = gVi \) and \( U = dgVi \) to upper bound the distance \( \|C_{Vi} - C_{gVi}\| \) by

\[
\text{bnd} = \text{bnd}(gVi, dgVi) = 2c \text{rdgVi} \left( 1 - \frac{M_{gVi}}{M_{dgVi}} \right) . \quad (14)
\]

This bound is computable with information in \( D^i \) only and can be used to guarantee that (11) holds by ensuring

\[
\|p' - C_{gVi}\| \geq \text{bnd} \quad (15)
\]

holds. The point \( p' \) to which agent \( i \) moves to is determined as follows: move towards \( C_{gVi} \) as much as possible in one timestep until it is within distance \( \text{bnd} \), 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 (14) grows larger and (15) 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.
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 less than or equal to this radius, including point-to-point communication, multi-hop communication, and sensing. For simplicity, we assume that agents have the capability to acquire this information for arbitrarily large $R_i$. Implicit in this model is the fact that larger radii correspond to more costly acquisition of information.

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

**Lemma 5.3.** Assume that at timestep $t_\ast \in \mathbb{Z}_{\geq 0}$, 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 $D_{\text{aff}}(t_\ast) = \{(p_i(t_\ast), 0), \ldots, (p_n(t_\ast), 0)\} \in (\mathbb{S} \times \mathbb{R}_{\geq 0})^n$ and $D_i = \{(p_i(t), q_i)\} \subset (\mathbb{S} \times \mathbb{R}_{\geq 0})^n$ be any vector whose $j$th component is $(p_j(t), q_j)$, $0 \leq j < n$. Then, $\pi_{A_i}(D^i)$ contains all and only the subset of agents in which $i$ can successfully compute correct guaranteed Voronoi cell. Let $D_{\text{aff}}(t_\ast) \subset (\mathbb{S} \times \mathbb{R}_{\geq 0})^n$ be any vector whose $j$th component is $(p_j(t_\ast), 0)$, $0 \leq j < n$. Then, $\pi_{A_i}(D^i)$ contains all and only the subset of agents in which $i$ can successfully compute correct guaranteed Voronoi cell. Let $D_{\text{aff}}(t_\ast) \subset (\mathbb{S} \times \mathbb{R}_{\geq 0})^n$ be any vector whose $j$th component is $(p_j(t_\ast), 0)$, $0 \leq j < n$. Then, $\pi_{A_i}(D^i)$ contains all and only the subset of agents in which $i$ can successfully compute correct guaranteed Voronoi cell. Let $D_{\text{aff}}(t_\ast) \subset (\mathbb{S} \times \mathbb{R}_{\geq 0})^n$ be any vector whose $j$th component is $(p_j(t_\ast), 0)$, $0 \leq j < n$. Then, $\pi_{A_i}(D^i)$ contains all and only the subset of agents in which $i$ can successfully compute correct guaranteed Voronoi cell.
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  that can no longer send or receive information to/from any other agent . Once all other agents have updated their information according to Voronoi cell computation, notice that for all the remaining agents , which continue to run the self-triggered centroid algorithm normally without agent . On the other hand if a new agent 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 .

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 , which corresponds to the composition of a “decide/acquire-up-to-date-information” map and a “move-and-update-uncertainty” map , i.e., . Our analysis strategy here is shaped by the fact that and, consequently, 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 , the agents’ position evolving under the self-triggered centroid algorithm from any initial network configuration in converges to the set of centroidal Voronoi configurations.

Since the map 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 , whose trajectories include the ones of and, apply the LaSalle Invariance Principle for set-valued maps, e.g., (Bullo et al., 2009).

Next, we define formally. For convenience, we recall that and that the elements of are referred to as , for each , to ease the exposition, we divide the construction of 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 , whose ith component is

where .

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 , associates the Cartesian product whose ith component is either (agent does not get any up-to-date information) or the vector

where , otherwise (agent gets updated information). Recall that is the set of neighbors of agent given the partition . It is not difficult to show that is closed (a set-valued map ).

We define the set-valued map , by . Given the continuity of and the closedness of , the map is closed. Moreover, if , is an evolution of the self-triggered centroid algorithm, then , is a trajectory of .

The next result establishes the monotonic evolution of the aggregate function along the trajectories of . With a slight abuse of notation, denote also by the extension of the aggregate function to the space , i.e., .

Lemma 6.2. is monotonically nonincreasing along the trajectories of .

Proof. Let and . For convenience, let and . To establish , we use the formulation (2) and divide our reasoning in two steps. First, we fix the partition . For each , if , then because agent does not move according to the definition of . If instead, , then, by Lemma 5.1 and Proposition 5.2, we have that , then, in either case, it follows from Lemma 2.1 that . Second, the optimality of the Voronoi partition stated in Lemma 2.1 guarantees that , and the result follows. □

One can establish the next result using Lemma 6.2 and the fact that is closed and its trajectories are bounded and belong to the closed set .

Lemma 6.3. Let be a trajectory of (16). Then, the -limit set belongs to , and is weakly positively invariant for , i.e., .

Proof. Let be a trajectory of (16). The fact that follows from being bounded. Let ; then there exists a subsequence which must have a convergent subsequence, i.e., there exists such that . By definition, , which implies that is weakly positively invariant. Now consider the sequence , since , since is bounded and is non-increasing along , on the sequence is decreasing and bounded from below, and therefore, convergent. Let satisfy . Take any ; accordingly, there exists a subsequence such that . Since is bounded, . Given , we conclude . □

We are finally ready to establish the asymptotic convergence of the self-triggered centroid algorithm.
Proof of Proposition 6.1. Let $\gamma = \{D(t_i)\}_{i \in \mathbb{Z}_{\geq 0}}$ be an evolution of the self-triggered centroid algorithm. Define $\gamma' = \{D'(t_i)\}_{i \in \mathbb{Z}_{\geq 0}}$ by $D'(t_i) = f_{\text{sync}}(D(t_i))$. Note that $\text{loc}(D(t_i)) = \text{loc}(D'(t_i))$. Since $\gamma'$ is a trajectory of $T_{\text{sync}}$, Lemma 6.3 guarantees that $\Omega(\gamma')$ is weakly positively invariant and belongs to $H^{-1}(c)$, for some $c \in \mathbb{R}$. Next, we show that $\Omega(\gamma') \subseteq \{D \in (S \times \mathbb{R})^n \mid \text{for } i \in \{1, \ldots, n\}, \|p_i - C_{\gamma_i}(\pi_{\gamma_i}(D))\| \leq \text{bnd}(\pi_{\gamma_i}(D'))\}$. (17)

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

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

$$\text{bnd}(\pi_{\gamma_i}(D')) \leq \max\{\|p_i - C_{\gamma_i}(\pi_{\gamma_i}(D))\|, \epsilon\},$$ (18)

for all $i \in \{1, \ldots, n\}$ and all $D \in \Omega(\gamma')$. Let us now show that $\Omega(\gamma') \subseteq \{D \in (S \times \mathbb{R})^n \mid \text{for } i \in \{1, \ldots, n\}, \|p_i - C_{\gamma_i}(\pi_{\gamma_i}(D))\| \leq \text{bnd}(\pi_{\gamma_i}(D'))\}$.

Consider $D \in \Omega(\gamma')$. Since $\Omega(\gamma')$ is weakly positively invariant, there exists $D_1 \in \Omega(\gamma') \cap T_{\text{sync}}(D)$. Note that (17) implies that $\text{loc}(D_1) = \text{loc}(D)$. We consider two cases, depending on whether or not agents have got up-to-date information in $D_1$. If agent $i$ gets up-to-date information, then $\text{bnd}(\pi_{\gamma_i}(D')) = 0$, and consequently, from (17), $p_i = p_i^* = C_{\gamma_i}(\pi_{\gamma_i}(D_1)) = C_{\gamma_i}$, and the result follows. If agent $i$ does not get up-to-date information, then $\text{bnd}(\pi_{\gamma_i}(D')) > \text{bnd}(\pi_{\gamma_i}(D_1))$ and $\|\pi_{\gamma_i}(\pi_{\gamma_i}(D'))\| < \|\pi_{\gamma_i}(\pi_{\gamma_i}(D_1))\|$ by Lemma 4.1. Again, using the fact that $\Omega(\gamma')$ is weakly positively invariant set, there exists $D_2 \in \Omega(\gamma') \cap T_{\text{sync}}(D_1)$. 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, $\|p_i - C_{\gamma_i}\| \to 0$ while $\text{bnd}$, monotonically increases towards $\text{diam}(S)$. For sufficiently large $\epsilon$, we have that $\|p_i - C_{\gamma_i}(\pi_{\gamma_i}(D'))\| < \epsilon$. Then (18) implies $\text{bnd}(\pi_{\gamma_i}(D')) < \epsilon$, which contradicts the fact that $\text{bnd}(\pi_{\gamma_i}(D'))$ tends to $\text{diam}(S)$. This ends the proof. $\square$

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 $\delta$, 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 $\delta$ of the set of centroidal Voronoi configurations. $\square$

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 $v_{\text{max}}$. 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 $v_{\text{max}}$ 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 $\nabla H$ 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 $\nabla H$ tends to zero. From (Bullo et al., 2009; Du et al., 1999), we know that $\frac{H}{H_{\text{max}}} = 2M_{AV}(p_i - C_{\gamma_i})$ for each $i \in \{1, \ldots, n\}$, and hence $\frac{\|\nabla H\|}{\|p_i - C_{\gamma_i}\| + \text{bnd}} \leq 2M_{AV}(\|p_i - C_{\gamma_i}\| + \text{bnd})$.

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 $\delta$, with $0 < \delta \ll 1$, $2M_{AV}(\|p_i - C_{\gamma_i}\| + \text{bnd}) < \delta$ (19)

for all $i \in \{1, \ldots, n\}$. This check can be implemented in a number of ways. Here, we use a convergencetcast 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 (19) is satisfied for $A_i \cup \{i\}$. 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 $i$ passes this message to its children or sends an acknowledgment to its parent if and only if (19) is satisfied for $j$. At the end of this procedure, the root $i$ has the necessary information to determine if (19) holds for all agents. If this is the case, agent $i$ broadcasts a message to all agents to set $\delta_{\text{max}} = v_{\text{max}}/2$ 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 & Tsitsiklis, 1997). Let $t^i = \{t_i^0, t_i^1, t_i^2, \ldots\} \subset \mathbb{R}_{\geq 0}$ be a time schedule for agent $i \in \{1, \ldots, n\}$. Assume agent $i$ executes the algorithm according to $t^i$, i.e., the agent executes the steps 1 - 12: described in Table 5 at time $t_i^j$, for $\epsilon \in \mathbb{Z}_{\geq 0}$, with timestep $(t_i^j - t_i^{j-1})$ instead of $\Delta t$. 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 synchroniza-

tion, see e.g., (Lin, Morse, & Anderson, 2007). Second, we use this to lay out a proof strategy similar to the one used for the synchronous case.

Analytic synchronization is a procedure that consists of merging together the individual time schedules $t^i, i \in \{1, \ldots, n\}$, of the network agents into a global time schedule $T = \{t, t_1, t_2, \ldots\}$ by setting $T = \cup_{i=1}^n t^i$. This synchronization is performed only for analysis purposes, i.e., $T$ is unknown to the individual agents. Note that more than one agent may be active at any given $t \in T$. For convenience, we define $\Delta t_i = t_{i+1} - t_i > 0$, for $\epsilon \in \mathbb{Z}_{\geq 0}$, i.e., $\Delta t_i$ is the time from $t_i$ until at least one agent is active again.

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 $\mathcal{T}_{\text{async}}$ whose trajectories include the asynchronous executions of the self-triggered centroid algorithm. As before, the construction of $\mathcal{T}_{\text{async}}$ 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 $\mathcal{T}_{\text{async}}$ 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 $(x, \ell) \in (S \times R_{>0})^n \times S \times Z_{\geq 0}$, where $x = ((D^1, u^1), \ldots, (D^n, u^n))$, $u^i$ denotes the waypoint of agent $i \in \{1, \ldots, n\}$ and $\ell$ is a time counter. For ease of notation, let

$$(S \times R_{>0})^n = ((S \times R_{>0})^n \times S) \times Z_{\geq 0}.$$ 

Motion and uncertainty update. The motion and time update map $\mathcal{M} : (S \times R_{>0})^n \to (S \times R_{>0})^n$ 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 $\mathcal{M}(x, \ell) = (\mathcal{M}_1(x, \ell), \ldots, \mathcal{M}_n(x, \ell), \ell)$ where

$$\mathcal{M}_i(x, \ell) = \left( (p_i^j, \min[r_i^j + \Delta t_i, \text{diam}(S)]), \ldots, (\text{tb}(p_i^j, v_{\text{max}} \Delta t_i, u^i, 0), 0), \ldots, (p_i^j, \min[r_i^j + \Delta t_i, \text{diam}(S)], u^i) \right).$$

Note that $\text{tb}(p_i^j, v_{\text{max}} \Delta t_i, u^i, 0)$ corresponds to where agent $i$ can get to in time $\Delta t_i$ 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 $\mathcal{U} : (S \times R_{>0})^n \to (S \times R_{>0})^{n'}$. 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 $i$th component, $i \in \{1, \ldots, n\}$, is one of the following three possibilities: either (i) the vector $(D^1, u^i)$, which means $i$ is not active at timestep $\ell$, (ii) the vector $(D^1, \text{tb}(p_i^j, v_{\text{max}}(\ell - t_i), C_{\mathcal{G}_i}, \text{bnd}_i))$, for some $\ell' > \ell$, with $C_{\mathcal{G}_i} = C_{\mathcal{G}_i}(\pi_i(D^1))$, $\text{bnd}_i = \text{bnd}_i(\pi_i(D^1))$, and $A_i' = \{i\} \cup \text{argmin}_{i \in \{1, \ldots, n\}}(|(p_i^j)|)^{1/2}$, which means $i$ is active at timestep $\ell$ and recomputes its waypoint but does not get any up-to-date information; or (iii) the vector

$$(p_1^i, r_1^i), \ldots, (p_n^i, r_n^i), \text{tb}(p_i^j, v_{\text{max}}(\ell - t_i), C_{\mathcal{G}_i}, \text{bnd}_i),$$

for some $\ell' > \ell$, with $C_{\mathcal{G}_i} = C_{\mathcal{G}_i}(\pi_i(D^1))$, $\text{bnd}_i = \text{bnd}_i(\pi_i(D^1))$, and $A_i' = \{i\} \cup \text{argmin}_{i \in \{1, \ldots, n\}}(|(p_i^j)|)^{1/2}$, which means $i$ is active at timestep $\ell$ and recomputes its waypoint but does not get any up-to-date information; or (iii) the vector

$$(p_1^i, r_1^i), \ldots, (p_n^i, r_n^i), \text{tb}(p_i^j, v_{\text{max}}(\ell - t_i), C_{\mathcal{G}_i}, \text{bnd}_i).$$

Finally, we define the set-valued map $\mathcal{T}_{\text{async}} : (S \times R_{>0})^n \to (S \times R_{>0})^n$ by $\mathcal{T}_{\text{async}} = \mathcal{U} \circ \mathcal{M}$. Given the continuity of $\mathcal{M}$ and the closedness of $\mathcal{U}$, the map $\mathcal{T}_{\text{async}}$ is closed. Moreover, the asynchronous executions of the self-triggered centroid algorithm with time schedules $\mathcal{T}', i \in \{1, \ldots, n\}$ are trajectories of

$$(x(\ell + 1), \ell + 1) \in \mathcal{T}_{\text{async}}(x(\ell), \ell).$$

Equipped with the definition of $\mathcal{T}_{\text{async}}$, 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 $\mathcal{T}', i \in \{1, \ldots, n\}$ are infinite and unbounded. For $e \in [0, \text{diam}(S)]$, the agents’ position evolving under the asynchronous self-triggered centroid algorithm with time schedules $\mathcal{T}', i \in \{1, \ldots, n\}$ 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 $4 \times 4$ square, with a maximum velocity $v_{\text{max}} = 1$ m/s. The synchronous executions operate with $\Delta t = 0.25$ s. In the asynchronous execution shown in Fig. 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 timesteps 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 (Fireuzabadi, 2007) for quantifying the total power $P_i$ used by agent $i$ to communicate, in dBmW power units,

$$P_i = 10 \log_{10} \left( \sum_{j=1}^{n} \beta^2 10^{0.1P_{i-j} + \sigma|p_i - p_j|} \right),$$

where $\alpha > 0$ and $\beta > 0$ depend on the characteristics of the wireless medium and $P_{i-j}$ 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.

Figs. 4 and 5 illustrate an execution of the self-triggered centroid algorithm for a density $\phi$ 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, Fig. 5 shows an execution that is also incorporating the distributed algorithm for decreasing velocity.

Fig. 6 shows the average communication power expenditure and the average time to convergence of the self-triggered centroid algorithm for varying $\epsilon$ over 20 random initial agent positions based on uniformly sampling the domain. One can see how as $\epsilon$ gets larger, the communication effort of the agents decreases at the cost of a slower convergence on the value of $H$. Interestingly, for small $\epsilon$, 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.

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. 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 by studying for how long agents can execute the proposed laws without fresh information, and explore the extension of these ideas to scenarios with limited-range interactions and other coordination tasks.

Acknowledgment

This work is supported by NSF Award CMMI-0908508.

References


