Balancing Sensing and Coverage in Mobile Sensor Networks: Aggregation Based Approach

Carlos H. Caicedo-N. and Miloš Žefran

Department of Electrical and Computer Engineering
University of Illinois at Chicago, Chicago, IL 60607
{ccaice1,mzefran}@uic.edu

Abstract—The paper studies how to assign mobile robots in a mobile wireless sensor network to different tasks that the network must perform so that each task receives its fair share of the resources in the network. In particular, we address the case when a set of autonomous mobile sensing robots is deployed inside a convex region, and the robots need to distribute themselves over the region uniformly, while also quickly locating one or more biochemical sources that might appear inside their region.

The main issue is then which robots should be assigned to the sensing task and move towards the chemical source, thereby diminishing the coverage of the region. The right balance between the sensing performance and coverage performance depends on the application in which the network is used. We thus propose an algorithm that given the area of the region and the number of agents in the network, makes the task assignment so that the number of sensing robots equals an arbitrary given function of these two parameters. We also show how these parameters can be estimated using aggregation algorithm as an alternative to less robust consensus protocols.

I. INTRODUCTION

Current advances in sensor technology have resulted in the mass production of small, energy-efficient and inexpensive micro-sensors. These small sensors can be mounted on mobile platforms, and typically have communication and computation capabilities. While each of them might not be as precise as its expensive counterparts, their low cost allows them to be deployed over a wide area, forming a reliable network that can monitor the region of interest and accurately detect important events.

Sensor networks are a vibrant area of research so we will limit literature review to contributions that are directly relevant to our work. The advantages, as well as some of the challenges that this new technology has to offer have been studied in [1], [2] and the references therein. In [3]–[5] the authors explore different communication protocols that can be used for such systems. How the network scales with respect to different measures is studied in [6], [7], while [8], [9] studies how to place the sensors to optimize the coverage of a region. Estimation of the location of a biochemical source with a sensor network is studied in [10].

Most of the papers on control of sensor network focus on a single task such as coverage [8], [11], [12], flocking [13]–[15], source localization [16], or rendez-vous [17], [18]; few authors address how to control the system when different tasks need to be performed. An exception is [19], where the authors frame the problem of a sensor network reaching different target points as they occur in the environment as a variation of the traveling salesman problem.

In [20] the problem of coordinating the coverage and sensing tasks in a mobile sensor network was first addressed, and two control algorithms were proposed. The algorithms did not scale well and depended on a number of parameters that had to be tuned to a particular task/environment. The main idea was to define a utility measure for each task that allowed each robot to decide which task to perform. The present paper generalizes the idea and proposes an algorithm that is distributed and thus scalable, while allowing to tune the performance of the network through a function of the area of the region of deployment and the number of agents. The function determines how any pair of utility functions assigned to the two tasks can be weighted so that the robots can decide which task to perform. The approach has the advantage that it can be readily generalized to more than two tasks, a topic of forthcoming publication. Another important contribution of the present paper is the effective use of aggregation algorithms instead of algorithms based on distributed consensus protocols in order to locally estimate the variables required for the distributed evaluation of the aforementioned weighting function. In particular, we show that aggregation algorithms are much more robust and have better convergence than those using consensus protocols.

II. PROBLEM STATEMENT

In traditional sensor networks (fire detectors in a building, weather instruments around the city, temperature sensors in a building, etc.), each sensor covers just a small portion of the whole environment. If a sensor fails, there is no way for the other sensors in the network to relocate to address the failure. Mobile sensor networks, as the name suggests, allow each sensor in the network to move depending on the state of the network and the environment. Furthermore, allowing the agents to communicate allows the network to perform more elaborated coordination tasks.

We are studying the problem where a group of mobile agents is deployed inside a convex region, and we want to obtain the best possible coverage of the region. At the same
time, the sensors should quickly respond to one or more events, for instance a biochemical source that might appear in the environment. We propose a distributed control algorithm that, given a weighting function that depends on the area of the region and the size of the network, determines how many agents should perform the coverage task, and how many agents should move in order to localize the source. Clearly, moving all the agents towards the event would make large portions of the region of interest vulnerable, while moving just one or two agents towards the event might not suffice to quickly locate the disturbance.

The distributed algorithm that we present allows the agents to determine the task they should perform based on a weighting function that determines the relative utility of an agent performing each individual task. However, in contrast to the most existing work in the area, we use the aggregation algorithm [21] rather than a modification of distributed consensus protocol [22] to estimate the number of agents in the network which are needed to compute the weighting function.

III. Model

We are interested in an algorithm such that given a surface $A$ and $n$ mobile agents, the agents will uniformly distribute over $A$. At the same time the agents should quickly locate (and possibly neutralize) one or more biochemical sources that could appear at any point inside $A$. Performing either task should not substantially compromise the other task.

We will assume that $A$ is a compact and convex surface in $\mathbb{R}^2$. Let $a_1, a_2, \ldots, a_n$ be the agents in the network, and let $p_i$ be the position of agent $a_i$ in $A$, expressed in some global coordinate system. We assume that the agent $a_i$ always knows its position $p_i$. Let $S$ be the source, and let $s$ be the source position. We assume that each agent has a sensor that at a distance $d$ from the source returns the following measurement:

$$S_d = \frac{K}{d^2 + 1} + N,$$

(1)

where $K$ is a constant that we will assume known in this paper and $N$ is a Gaussian measurement noise with zero mean and variance $\sigma^2$. This model is inspired by diffusion sources and we refer the reader to [23] for further information.

Define the Voronoi region $Q_i$ generated by the agent $a_i$ as a subset $Q_i \subset A$ given by

$$Q_i = \{ q \in A : ||q - p_i|| \leq ||q - p_j||, j \in \mathbb{N}, 1 \leq j \leq n \}.$$

(2)

In this paper we will not go through the properties of the Voronoi regions. For information on that subject we refer the reader to [24], [25].

We say that the agents $i$ and $j$ are neighbors in the sense of Voronoi if $Q_i \cap Q_j \neq \emptyset$. In other words, two agents are neighbors if their Voronoi regions have a common edge. We refer to such pair of agents simply as neighbors. Define $N_i \subset \mathbb{N}$ as the set of indices of all the neighbors of agent $i$

$$N_i = \{ j \in \mathbb{N} | a_j \text{ is a neighbor of } a_i \}.$$

We assume that $a_i$ has the knowledge of $N_i$ and can communicate with every neighbor $a_j, j \in N_i$ regardless of the distance between $a_i$ and $a_j$. In other words, the only condition for two agents to be able to communicate is that they share at least one point among their Voronoi regions. Note that this implies that the communication graph (where each agent represents a vertex, and there is an edge between two vertices if and only if the two agents are capable of communicating) is always connected.

For each agent we define one of two possible states: each agent $a_i$ is either doing sensing (state 1) or doing coverage (state 2). Assume the sensor location is described by the following first order dynamics

$$\dot{p}_i = u_i,$$

where $u_i$ is the motion control input for $a_i$. We will study the case when the sensor motion is governed by the following control laws:

$$u_i = \begin{cases} -k_1 (p_i - s_i), & k_1 > 0 \text{ if state is 1}, \\ -k_2 (p_i - C_i), & k_2 > 0 \text{ if state is 2}. \end{cases}$$

(3)

where $s_i$ is the estimate of the source location $s$ generated through a consistent estimator by agent $a_i$ using its measurements and the measurements of its neighbors, and $C_i$ is the centroid of its Voronoi region.

IV. Description of the Algorithm

Since the total area and the number of agents in the formation defines the quality of the coverage, and hence how many agents we can take away from this task without seriously compromising it, we define a weighting function $g_1 : \mathbb{R} \times \mathbb{Z} \rightarrow \mathbb{R}$ which, given the area of $A$ and the number of agents in the formation $n$, assigns the total number of agents performing the sensing task, according to some rule. We assume that the function $g_1$ is known by all the agents.

Using the weighting function $g_1$ allows us to abstract the implementation details for different application scenarios while at the same time providing enough flexibility to tune the network. It is easy to envision hierarchical control schemes where $g_1$ is tuned by a higher level supervisory controller depending on the state of the network; as long as the changes are slow enough compared to the dynamics of the network they do not affect the stability of the system. For example, the system can jump from all the agents doing overage, to the case where a fixed number of them or a percentage of them performs sensing.

We next describe the control algorithm. In particular, we will show that the algorithm makes the agents converge to an equilibrium where both sensing and coverage tasks are completed. We will also describe a robust distributed algorithm to compute the number of agents $n$, which is necessary for the evaluation of $g_1$.

Let $A$ be the numerical area, in some pre-defined units, of $A$. Let $M = g_1(A, n)$. Since the agents are deployed inside $A$ it is reasonable to assume that the agents know the value of $A$. Let $M_i$ be the value of $g_1(A, n_i)$ evaluated by the $i$-th agent,
based on \( n_i \), the number of agents that the agent \( a_i \) estimates in the network. We will describe a distributed algorithm for computing \( n_i \) in section V. Also, recall that \( g_1 (A, n) \) is that number of agents that should engage in the sensing task.

Given the sensor model (1), it is clear that sensing is best performed by the agents closer to the source. Each agent \( a_i \) can thus decide whether to do sensing only based on the information it receives from its neighbors and \( M_i \); if there are already at least \( M_i \) agents that are closer to the source than itself, \( a_i \) should do coverage; otherwise it should do sensing. The crucial piece of information needed by \( a_i \) to make this decision is thus how many agents already do sensing.

One way to provide global information in a sensor network is through broadcasting. But the amount of energy required for broadcasting does not scale favorably with \( A \), and it might be physically impossible for the agents to broadcast a message to every other agent in the formation. On the other hand, the sensing task attracts the agents towards a single location, so messages from sensing agents are naturally broadcast to all the other sensing agents. It is reasonable to assume that every sensing agent that is within a certain radius \( R \) of any other sensing agent, will be able to identify the presence (although not necessarily the exact location) of that other robot. Therefore, if all the sensing agents are inside a circle of radius \( R/2 \) about the source, then all of the sensing agents will detect the presence of each other, and hence, they will know how many sensing agents there are. As will be shown later, since the sensing agents converge toward the source, eventually they will all be inside such a circle and the count induced by the broadcast will be accurate.

Note that each agent \( a_i \) that has a neighbor which is doing sensing and is within \( R/2 \) of the source will know the number of agents doing sensing that are closer to the source than it is. If that number is greater than or equal to \( M_i \), that agent will go to coverage, otherwise, it will go to sensing. If an agent does not have any neighbor doing sensing they will not have the count of the sensing agents, so they have to rely only on the information that its neighbors have. We thus use a flag that we call \( \text{tsk-comp} \), task completed, that gets propagated through the network and we will show that in this way each agent is correctly assigned to a task.

The algorithm is described in table I.

### A. Analysis of the protocol

Since the source has to belong to one of the Voronoi regions\(^2\) every agent in the formation is either the closest to the source, or has a neighbor that is closer to the source. This is important because it ensures the correct propagation of the \( \text{tsk-comp} \) flag. If there would be an agent such that the source is not inside its Voronoi region, but has no neighbor which is closer to the source, then this agent would have to go to sensing, although there might be no need for it, and will mislead its neighbors.

\(^2\)In the degenerate case when the source is on the boundary of several Voronoi regions we can resolve the ambiguity by assigning unique IDs to the agents and choosing the one with the smallest ID.
Since the communication graph is connected, that flag will know that there are already M agents doing sensing. This guarantees that at least M of the sensing agents will be in the interior of a circle of radius \(th/2\) about the real source location, and our assumption of the agent being capable of hearing M agents by increasing the communication radius by \(th\) is justified.

Now, when agent \(a_{M+1}\) sees \(M\) agents doing sensing, it will go to coverage and will set its own “task completed” flag to 1, so its neighbors, which are further away from the source, will know that there are already \(M\) agents doing sensing. Since the communication graph is connected, that flag will be transmitted in at most \(d\) steps, where \(d\) is the diameter of the graph, to all the agents that are not doing sensing.

Therefore, the sensor network will converge to a state on which \(M\) agents are doing sensing, and \(n-M\) are doing coverage as stated.

Now we need to prove that the \(M\) agents doing sensing will, in fact, converge to the source and that the \(n-M\) agents will converge to the centroids of its respective Voronoi regions. As was mentioned in the previous proof, since we are considering a consistent estimator, \(s_i(t)\) will converge to \(s\). We will show that this will guarantee that the agent \(a_i\) will converge to \(s\).

**Proposition 3:** Once the \(M\) agents are in an \(\epsilon\) neighborhood of \(S\), the remaining \(n-M\) agents will converge to the centroids of its Voronoi regions.

**Proof:**

The proof follows the argument for convergence presented in [8]. Suppose \(a_i\) is an agent doing sensing. Since \(a_i\) converges to \(s\), then \(\lim_{t\to\infty} \hat{s}_i(t) = 0\), which implies that, for any \(\epsilon > 0\), there is a \(T\) such that whenever \(t > T\), \(|\hat{s}_i(t)| < \epsilon\).

Now consider

\[
J_c = \sum_{i=1}^{n} \int_{Q_i} ||q - p_i||^2 dq.
\]

Using the parallel axis theorem, it is shown that \(\partial J_c/\partial p_i\) can be written as

\[
\frac{\partial J_c}{\partial p_i} = 2M_{Q_i} (p_i - C_i),
\]

where \(M_{Q_i}\) is the mass of the region \(Q_i\).

Let \(T_1\) the set of indices that correspond to the agents that are doing coverage, and \(T_2\) the set of indices that correspond to the agents that are doing sensing, therefore

\[
J_c = \sum_{i=1}^{n} \int_{Q_i} ||q - p_i||^2 dq = \sum_{i \in T_1} \int_{Q_i} ||q - p_i||^2 dq + \sum_{j \in T_2} \int_{Q_j} ||q - p_j||^2 dq
\]

Since

\[
\frac{dJ_c}{dt} = \sum_{i=1}^{n} \frac{\partial J_c}{\partial p_i} \dot{p}_i
\]

then

\[
\frac{dJ_c}{dt} = \sum_{i \in T_1} 2M_{Q_i} (p_i - C_i) \dot{p}_i + \sum_{j \in T_2} 2M_{Q_j} (p_j - C_j) \dot{p}_j.
\]

The previous equation relates functions of the time parameter \(t\). Given that \(A\) is compact and \(M_{Q_i}(p_i - C_i)\) is continuous, it has an upper bound \(B\). Therefore,

\[
\frac{dJ_c}{dt} \leq B \sum_{i \in T_1} |\dot{p}_i| - k_2 \sum_{j \in T_1} 2M_{Q_j} (p_j - C_j)^2,
\]

and since the number of agents doing sensing is finite, we can find a time \(T\) such that, for any \(\epsilon > 0\) and for any agent \(a_i\) doing sensing, if \(t > T\), then \(|\dot{p}_i(t)| < \epsilon/(B|M_{T_1})\), so we get:

\[
\frac{dJ_c}{dt} \leq -k_2 \sum_{j \in T_1} 2M_{Q_j} (p_j - C_j)^2.
\]

Hence, the agents doing coverage will converge to circles of radius \(\sqrt{\epsilon}/(2k_2M_{Q_i})\) around their respective centroids. Since \(\epsilon\) is arbitrary, the conclusion follows.

The algorithm can, in fact, be extended to the case where \(N\) sources are present in the environment. Figure 2 presents the simulation result after implementing the algorithm when 2 sources appear in the workspace. The details of the generalization of the algorithm will be presented in forthcoming publications.

**V. ESTIMATING THE TOTAL NUMBER OF AGENTS: THE AGGREGATION ALGORITHM**

It should be noted that a key element in our algorithm is to know both \(A\), the area of \(A\) and \(n\), the total number of agents. Since the agents know where they are deployed, it is reasonable to assume they know \(A\). The problem of computing \(g_1(A, n)\) thus reduces to the estimation of \(n\). In this section, we present a distributed algorithm for this estimation.

The algorithm that we are going to consider is a minor modification of the one proposed in [21]. The method that is usually employed to solve this problem is to chose a variation of the so-called consensus protocol, itself the focus of much research in the control community (see for instance [22], [26], [27] among others). Remarkably enough, different consensus protocols were first studied in a more general setting about 20 years ago in the context of parallel computing [28]–[30]. Furthermore, aggregation (or pairwise averaging), which has received a fair share of attention in the last years in the context of gossiping algorithms [31] can also be seen as a special case of the results in [28].

Averaging algorithms based on distributed consensus protocols have several drawbacks. Most require an overall synchronization signal in the network that will allow all of the agents to update their values simultaneously. If there is a single agent that takes longer to compute its new value, the final result will be invalid. Furthermore, these algorithms are not robust with respect to changes in the number of agents; also, some of them require for their discrete-time implementation some a-priori knowledge about the size of the network in order to guarantee convergence. The aggregation algorithm presented in [21] only
requires synchronization between the pair of agents when they are communicating and with a minor modification, which will be shown next, it can made to be robust to changes in the size of the network.

In this protocol, each agent has a variable associated to it. It then chooses at random one of its neighbors and both agents update the values of their variables to their average. Repeating this process, the values associated to each agent will converge to the same number. Since the sum of the variables remains constant throughout the process, then the values will converge to the mean of the initial sum. The proof for convergence of the original algorithm can be found in either in [21] for this particular algorithm or in [28] for the more general setting, although we will present an alternative proof using control-theory tools.

The aggregation algorithm is used to estimate the total number of agents in the network as follows. Initially an agent $a_i$ is initialized with variable 1 and all the other agents start at 0. This can be implemented before deploying the robots in the area of interest, so it is a reasonable assumption. Each agent $a_i$ will choose, among its set of neighbors, the agent $a_j$ such that it differs the most with. If there are two or more neighbors that maximize the difference, it chooses one of them at random. The value associated to each agent will converge to $1/n$, where $n$ is the total number of agents in the formation, allowing in this way the distributed counting of robots.

3Alternatively, one could employ one of the distributed leader election algorithms [32].
Let \( x_r \) be the variable assigned to agent \( a_r \). Each agent \( a_r \) will chose among its neighbors the agent \( a_s \) for which \( |x_r - x_s| \) is maximum, and then they will update their values of \( x_r \) and \( x_s \) respectively to \( \frac{x_r + x_s}{2} \). Note that \( \sum x_i \) is constant; therefore, so is its mean \( \mu \).

Lemma 4: After two agents \( a_r, a_s \) update their values \( x_r, x_s \), which are supposed to be different, the value of \( V = \sum (x_i - \mu)^2 \) decreases.

Proof: Just note that \( V \) decreases as we update the values associated to the agents:

\[
(x_r - \mu)^2 + (x_s - \mu)^2 - 2 \left( \frac{x_r + x_s}{2} - \mu \right)^2 = \frac{1}{2} (x_r - x_s)^2
\]

which is greater than zero, because \( x_r \neq x_s \) from hypothesis. Therefore after updating the values of \( x_r \) and \( x_s \), \( V \) decreases as expected.

Proposition 5: The variables \( x_i \) will evolve to \( \mu \).

Proof: From lemma 4, and since the network can be represented by a connected graph, we know that if not all the variables \( x_i \) are equal among them then there should exist two neighbor agents that will decrease the value of \( V \). From LaSalle’s principle, the \( x_i \) will converge to the largest invariant set on which they are defined; which happens to be the set on which each \( x_i = \mu \).

It should be noted that this consensus algorithm is entirely asynchronous. The reader interested in synchronous consensus algorithms for both continuous and discrete-time systems can easily find information about them in the literature, for example [22], [26], [27], [30] among others.

A. Adding and removing agents from the formation

There might be situations on which the number of agents originally deployed in the network is insufficient for the task in hand, and more robots need to be added to satisfy the requirements. The agents that are already deployed in the formation do not need to know before hand how many new agents are going to be added to the system in case they are required, so they need to implement a way to count them. Also, agents might simply start to malfunction and, in this scenario, the neighboring agents need to be aware of excluding that agent from the formation while notifying all the other agents of the new situation. In this section, we will show how this two tasks can be implemented in a distributed fashion, which will allow the network to adapt to changes in the number of robots.

1) Adding agents to the formation: If \( r \) agents are added to the system, the consensus variable will now change from \( 1/n \) to \( 1/(n + r) \). Introducing the new agents with their associated value at 0, will allow the system to converge to the new required value. When the new agents arrive to the formation, their value (0) would be different from the ones that were already in consensus and would be included in the counting process as a consequence of this. It should be clear, repeating the argument of the previous section that the new values will converge to \( 1/(n + r) \) as expected.

2) Removing faulty agents from the formation: In a wireless sensor network, the agents can be deployed in a hostile environment or run out of battery. In both cases the chance of some agents starting to malfunction is a real possibility. We will assume that each agent is capable of recognizing which ones, if any, of its neighbors are malfunctioning and excluding them from any operation inside the network. Nonetheless, the information of an agent malfunctioning should be propagated in some way to all the agents in the formation. Here we present an algorithm that will allow to transmit this information throughout the network.

We will call session the set of events that occur between the last time one agent malfunctioning was detected and the new appearance of a faulty robot. The system starts with session 0 and will stay in session 0 until an agent detects a problem with at least one of its neighbors. When a faulty agent is detected, the agent will chose a random number \( r \) and will add it to the previous session number. Then, it will reset its value to one and will propagate the message informing its value and the new session number.

When this message arrives to another agent, the agent will compare the session number with the one that was already in progress. If the new session number is higher, then it will drop its previous value, will reset it to zero, and will perform the averaging operation of the aggregation algorithm. Then it will chose among its neighbors either the one that is still in the previous session, or the one in the new session for which the difference between the two values is maximal. If more than one agent is in some of those situations, it will chose which one to pick at random. If two agents decide to reset their values and initiate new sessions at the same time, the session with the highest code will prevail.

We claim that each agent will be included into the new session. Suppose there is at least one agent that hasn’t been included in the new session. Since the communication graph for the system is connected, there is a path between any agent that is not in the new session and the agent that started it. Therefore, there is at least one agent in the path that is not in the new session while one of its neighbors is. The neighbor agent that is in the session has a finite number of neighbors and since it has at least one neighbor that still is in the previous session, each time this neighbor communicates with its neighbors it chooses a neighbor in the old session to do the update. Therefore, the number of agents in the old session is always decreasing and hence eventually all the agents will be in the updated session.

It should be noted that changes in the communication topology do not affect the above discussion because the underlying communication graph will still be connected.

VI. SIMULATIONS

In order to verify the theoretical results, we simulated the system using Mathematica using the code originally written for [8]. The simulations show that the expected final positions for the system when 1 and 2 sources were deployed agree with the theoretical analysis. Furthermore, the performance criteria
evolve as expected. In simulations, 18 agents were deployed over the same surface $A$ in both cases. The initial and final positions of the agents in both cases, as well as the coverage cost $J_c$, are presented.

During the simulations, 18 agents were placed randomly in a convex surface $A$. The agents were initially allowed to perform just the coverage task, and then the source appeared in $A$.

Figure 1 shows the initial position, final position and evolution of the cost induced by the coverage function $J_c$, as the formation evolves. The coverage cost $J_c$ decreases although there are some minor spikes in it due to the measurement noise of the sensing agents. These spikes in $J_c$ can be better seen in Figure 2 which depicts the case where two sources need to be localized. After the agents in Figure 2 converge, the agents doing sensing are distributed between the 2 sources according to the values of $g_1$ and $g_2$ (5 and 3 for this example). The evolution of $J_c$ reflects the behavior of the system. It is decreasing until the agents detect the source and start moving to it. As before, $J_c$ decreases except for the spikes due to the measurement noise, which coincides with the behavior predicted analytically. The noisy estimates make the agents oscillate around the source. From our proof of Theorem 3 it follows that if the agents are oscillating in and out of the source, then we will have bounded peaks in $J_c$, which is reflected in the simulation.

VII. CONCLUSIONS

An algorithm that assigns mobile robots in a mobile wireless sensor network to different tasks so that each task receives its fair share of the resources has been presented. Given the area of the region and the number of agents in the network, the task assignment is made so that the number of sensing robots equals an arbitrary given function of these two parameters. The algorithm has been proven to be stable, distributed, scalable and asynchronous. While we assume that the area of the region is given, the aggregation algorithm [21] is employed to compute the number of agents in the network. An alternative control-theoretic proof for the convergence of the algorithm is also presented. The aggregation algorithm is shown to be superior to the commonly used algorithms based on the distributed consensus protocol, since it can be guaranteed to converge without a global synchronism in the network.

REFERENCES


