Distributed Algorithms for Robotic Networks

Francesco Bullo[1] Jorge Cortés[2] Sonia Martínez[2]

[1] Department of Mechanical Engineering, University of California, Santa Barbara, California 93106, bullo@engineering.ucsb.edu

[2] Department of Mechanical and Aerospace Engineering, University of California, San Diego, California 92093, {soniamd,cortes}@ucsd.edu

Contents

Glossary

Cooperative control: In recent years, the study of groups of robots and multi-agent systems has received a lot of attention. This interest has been driven by the envisioned applications of these systems in scientific and commercial domains. From a systems and control theoretic perspective, the challenges in cooperative control revolve around the analysis and design of distributed coordination algorithms that integrate the individual capabilities of the agents to achieve a desired coordination task.

Distributed algorithm: In a network composed of multiple agents, a coordination algorithm specifies a set of instructions for each agent that prescribe what to sense, what to communicate and to whom, how to process the information received, and how to move and interact with the environment. In order to be scalable, coordination algorithms need to rely as much as possible on local interactions between neighboring agents.

Complexity measures: Coordination algorithms are designed to enable networks of agents achieve a desired task. Since different algorithms can be designed to achieve the same task, performance metrics are necessary to classify them. Complexity measures provide a way to characterize the properties of coordination algorithms such as completion time, cost of communication, energy consumption, and memory requirements.

Averaging algorithms: Distributed coordination algorithms that perform weighted averages of the information received from neighboring agents are called averaging algorithms. Under suitable connectivity assumptions on the communication topology, averaging algorithms achieve agreement, i.e., the state of all agents approaches the same value. In certain cases, the agreement value can be explicitly determined as a function of the initial state of all agents.

Leader election: In leader election problems, the objective of a network of processors is to elect a leader. All processors have a variable "leader" initially set to unknown. The leader-election task is solved when only one processor has set the variable "leader" to `true`, and all other processors have set it to `false`.

LCR algorithm: The classic Le Lann-Chang-Roberts (LCR) algorithm solves the leader election task on a static network with the ring communication topology. Initially, each agent transmits its unique identifier to its neighbors. At each communication round, each agent compares the largest identifier received from other agents with its own identifier. If the received identifier is larger than its own, the agent declares itself a non-leader, and transmits it in the next communication round to its neighbors. If the received identifier is smaller than its own, the agent does nothing. Finally, if the received identifier is equal to its own, it declares itself a leader. The LCR algorithm achieves leader election with linear time complexity and quadratic total communication complexity, respectively.

Agree-and-pursue algorithm: Coordination algorithms for robotic networks combine the features of distributed algorithms for networks of processors with the sensing and control capabilities of the robots. The agree-and-pursue motion coordination algorithm is an example of this fusion. Multiple robotic agents

moving on a circle seek to agree on a common direction of motion while at the same achieving an equally-spaced distribution along the circle. The agree-and-pursue algorithm achieves both tasks combining ideas from leader election on a changing communication topology with basic control primitives such as "follow your closest neighbor in your direction of motion."

1 Definition of the Subject and Its Importance

The study of distributed algorithms for robotic networks is motivated by the recent emergence of low-power, highly-autonomous devices equipped with sensing, communication, processing, and control capabilities. In the near future, cooperative robotic sensor networks will perform critical tasks in disaster recovery, homeland security, and environmental monitoring. Such networks will require efficient and robust distributed algorithms with guaranteed quality-of-service. In order to design coordination algorithms with these desirable capabilities, it is necessary to develop new frameworks to design and formalize the operation of robotic networks and novel tools to analyze their behavior.

2 Introduction

Distributed algorithms are a classic subject of study for networks composed of individual processors with communication capabilities. Within the automata-theoretic literature, important research topics on distributed algorithms include the introduction of mathematical models and precise specifications for their behavior, the formal assessment of their correctness, and the characterization of their complexity.

Robotic networks have distinctive features that make them unique when compared with networks of static processors. These features include the operation under ad-hoc dynamically changing communication topologies and the complexity that results from the combination of continuous- and discrete-time dynamics. The spatially-distributed character of robotic networks and their dynamic interaction with the environment make the classic study of distributed algorithms, typically restricted to static networks, not directly applicable.

This chapter brings together distributed algorithms for networks of processors and for robotic networks. The first part of the chapter is devoted to a formal discussion about distributed algorithms for a synchronous network of processors. This treatment serves as a brief introduction to important issues considered in the literature on distributed algorithms such as network evolution, task completion, and complexity notions. To illustrate the ideas, we consider the classic Le Lann-Chang-Roberts (LCR) algorithm, which solves the leader election task on a static network with the ring communication topology. Next, we present a class of distributed algorithms called averaging algorithms, where each processor computes a weighted average of the messages received from its neighbors. These algorithms can be described as linear dynamical systems, and

their correctness and complexity analysis has nice connections with the fields of linear algebra and Markov chains.

The second part of the chapter presents a formal model for robotic networks that explicitly takes into account communication, sensing, control, and processing. The notions of time, communication, and space complexity introduced here allow us to characterize the performance of coordination algorithms, and rigorously compare the performance of one algorithm versus another. In general, the computation of these notions is a complex problem that requires a combination of tools from dynamical systems, control theory, linear algebra, and distributed algorithms. We illustrate these concepts in three different scenarios: the agree-and-pursue algorithm for a group of robots moving on a circle, aggregation algorithms that steer robots to a common location, and deployment algorithms that make robots optimally cover a region of interest. In each case, we report results on the complexity associated to the achievement of the desired task. The chapter ends with a discussion about future research directions.

3 Distributed Algorithms on Networks of Processors

Here we introduce a synchronous network as a group of processors with the ability to exchange messages and perform local computations. What we present is a basic classic model studied extensively in the distributed algorithms literature. Our treatment is directly adopted with minor variations from the texts [1] and [2].

3.1 Physical Components and Computational Models

Loosely speaking, a synchronous network is a group of processors, or nodes, that possess a local state, exchange messages among neighbors, and compute an update to their local state based on the received messages. Each processor alternates the two tasks of exchanging messages with its neighboring processors and of performing a computation step.

Let us begin by providing some basic definitions. A *directed graph* [3], in short *digraph*, of order n is a pair $G = (V, E)$ where V is a set with n elements called *vertices* (or sometimes *nodes*) and E is a set of ordered pair of vertices called *edges*. In other words, $E \subseteq V \times V$. We call V and E the *vertex set* and *edge set*, respectively. For $u, v \in V$, the ordered pair (u, v) denotes an edge *from* u *to* v. The vertex u is called an *in-neighbor* of v, and v is called an *out-neighbor* of u. A *directed path* in a digraph is an ordered sequence of vertices such that any two consecutive vertices in the sequence are a directed edge of the digraph. A vertex of a digraph is *globally reachable* if it can be reached from any other vertex by traversing a directed path. A digraph is *strongly connected* if every vertex is globally reachable.

A *cycle* in a digraph is a non-trivial directed path that starts and ends at the same vertex. A digraph is *acyclic* if it contains no cycles. In an acyclic graph,

every vertex with no in-neighbors is named *source*, and every vertex with no out-neighbors is named *sink*. A *directed tree* is an acyclic digraph with the following property: there exists a vertex, called the root, such that any other vertex of the digraph can be reached by one and only one path starting at the root. A *directed spanning tree*, or simply a *spanning tree*, of a digraph is a subgraph that is a directed tree and has the same vertex set as the digraph. A *directed chain* is a directed tree with exactly one source and one sink. A *directed ring digraph* is the cycle obtained by adding to the edge set of a chain a new edge from its sink to its source. Figure 1 illustrates these notions.

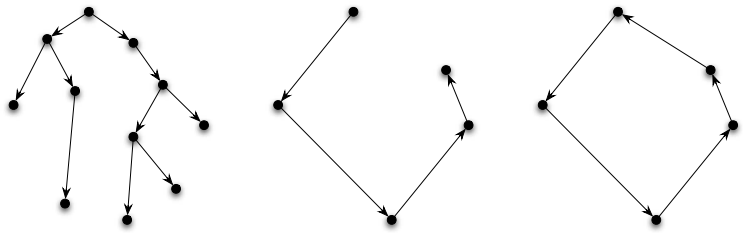


Figure 1: From left to right, directed tree, chain, and ring digraphs.

The physical component of a synchronous network $\mathcal{S}$ is a digraph (I, E_{cmm}), where

(i) $I = \{1, \ldots, n\}$ is called the *set of unique identifiers (UIDs)*, and

(ii) E_{cmm} is a set of directed edges over the vertices $\{1, \ldots, n\}$, called the communication links.

The set E_{cmm} models the topology of the communication service among the nodes: for $i, j \in \{1, \ldots, n\}$, processor i can send a message to processor j if the directed edge (i, j) is present in E_{cmm}. Note that, unlike the standard treatments in [1] and [2], we do not assume the digraph to be strongly connected; the required connectivity assumption is specified on a case by case basis.

Next, we discuss the state and the algorithms that each processor possesses and executes, respectively. By convention, we let the superscript $[i]$ denote any quantity associated with the node i. A *distributed algorithm* $\mathcal{DA}$ for a network $\mathcal{S}$ consists of the sets:

(i) $\mathbb{A}$, a set containing the `null` element, called the *communication alphabet*; elements of $\mathbb{A}$ are called *messages*;

(ii) $W^{[i]}$, $i \in I$, called the *processor state sets*;

(iii) $W_0^{[i]} \subseteq W^{[i]}$, $i \in I$, sets of *allowable initial values*;

and of the maps:

(i) $\text{msg}^{[i]} : W^{[i]} \times I \to \mathbb{A}$, $i \in I$, called *message-generation functions*;

(ii) $\text{stf}^{[i]} : W^{[i]} \times \mathbb{A}^n \to W^{[i]}$, $i \in I$, called *state-transition functions.*

If $W^{[i]} = W$, $\text{msg}^{[i]} = \text{msg}$, and $\text{stf}^{[i]} = \text{stf}$ for all $i \in I$, then $\mathcal{DA}$ is said to be *uniform* and is described by a tuple $(\mathbb{A}, W, \{W_0^{[i]}\}_{i \in I}, \text{msg}, \text{stf})$.

Now, with all elements in place, we can explain in more detail how a synchronous network executes a distributed algorithm. The *state* of processor i is a variable $w^{[i]} \in W^{[i]}$, initially set equal to an allowable value in $W_0^{[i]}$. At each time instant $\ell \in \mathbb{Z}_{\geq 0}$, processor i sends to each of its out-neighbors j in the communication digraph (I, E_{cmm}) a message (possibly the `null` message) computed by applying the message-generation function $\text{msg}^{[i]}$ to the current values of its state $w^{[i]}$ and to the identity j. Subsequently, but still at time instant $\ell \in \mathbb{Z}_{\geq 0}$, processor i updates the value of its state $w^{[i]}$ by applying the state-transition function $\text{stf}^{[i]}$ to the current value of its state $w^{[i]}$ and to the messages it receives from its in-neighbors. Note that, at each round, the first step is transmission and the second one is computation.

We conclude this section with two sets of remarks. We first discuss some aspects of our communication model that have a large impact on subsequent development. We then collect a few general comments about distributed algorithms on networks.

Remarks 3.1 (Aspects of the communication model) *(i) The network $\mathcal{S}$ and the algorithm $\mathcal{DA}$ are referred to as* synchronous *because the communications between all processors takes place at the same time for all processors.*

(ii) Communication is modeled as a so-called "point to point" service: a processor can specify different messages for different out-neighbors and knows the processor identity corresponding to any incoming message.

(iii) Information is exchanged between processors as messages, i.e., elements of the alphabet $\mathbb{A}$; the message `null` *indicates no communication. Messages might encode logical expressions such as* `true` *and* `false`*, or finite-resolution quantized representations of integer and real numbers.*

(iv) In some uniform algorithms, the messages between processors are the processors' states. In such cases, the corresponding communication alphabet is $\mathbb{A} = W \cup \{\texttt{null}\}$ and the message generation function $\text{msg}_{\text{std}}(w, j) = w$ is referred to as the standard message-generation function. •

Remarks 3.2 (Advanced topics: Control structures and failures) *(i) Processors in a network have only partial information about the network topology. In general, each processor only knows its own UID, and the UID of its in- and out-neighbors. Sometimes we assume that the processor knows the network diameter. In some cases [2], actively running networks might depend upon "control structures," i.e., structures that are computed at initial time and are exploited in subsequent algorithms. For example, routing tables might be computed for routing problems, "leader" processors*

might be elected and tree structures might be computed and represented in a distributed manner for various tasks, e.g., coloring or maximal independent set problems. We present some sample algorithms to compute these structures below.

(ii) A key issue in the study of distributed algorithms is the possible occurrence of failures. A network might experience intermittent or permanent communication failures: along given edges a `null` *message or an arbitrary message might be delivered instead of the intended value. Alternatively, a network might experience various types of processor failures: a processor might transmit only* `null` *messages (i.e., the* msg *function returns* `null` *always), a processor might quit updating its state (i.e., the* stf *function neglects incoming messages and returns the current state value), or a processor might implement arbitrarily modified* msg *and* stf *functions. The latter situation, in which completely arbitrary and possibly malicious behavior is adopted by faulty nodes, is referred to as a Byzantine failure in the distributed algorithms literature.* •

3.2 Complexity Notions

Here we begin our analysis of the performance of distributed algorithms. We introduce a notion of algorithm completion and, in turn, we introduce the classic notions of time, space, and communication complexity.

We say that an algorithm *terminates* when only `null` messages are transmitted and all processors states become constants.

Remark 3.3 (Alternative termination notions) *(i) In the interest of simplicity, we have defined evolutions to be unbounded in time and we do not explicitly require algorithms to actually have termination conditions, i.e., to be able to detect when termination takes place.*

(ii) It is also possible to define the termination time as the first instant when a given problem or task is achieved, independently of the fact that the algorithm might continue to transmit data subsequently. •

The notion of time complexity measures the time required by a distributed algorithm to terminate. More specifically, the *(worst-case) time complexity* of a distributed algorithm $\mathcal{DA}$ on a network $\mathcal{S}$, denoted $\mathrm{TC}(\mathcal{DA}, \mathcal{S})$, is the maximum number of rounds required by the execution of $\mathcal{DA}$ on $\mathcal{S}$ among all allowable initial states until termination.

Next, we quantify memory and communication requirements of distributed algorithms. From an information theory viewpoint [4], the information content of a memory variable or of a message is properly measured in bits. On the other hand, it is convenient to use the alternative notions of "basic memory unit" and "basic message." It is customary [2] to assume that a "basic memory unit" or a "basic message" contains $\log(n)$ bits so that, for example, the information content of a robot identifier $i \in \{1, \ldots, n\}$ is $\log(n)$ bits or, equivalently, one

"basic memory unit." Note that elements of the processor state set W or of the alphabet set $\mathbb{A}$ might amount to multiple basic memory units or basic messages; the `null` message has zero cost. Unless specified otherwise, the following definitions and examples are stated in terms of basic memory unit and basic messages.

- The *(worst-case) space complexity* of a distributed algorithm $\mathcal{DA}$ on a network $\mathcal{S}$, denoted by $\mathrm{SC}(\mathcal{DA}, \mathcal{S})$, is the maximum number of basic memory units required by a processor executing the $\mathcal{DA}$ on $\mathcal{S}$ among all processors and among all allowable initial states until termination.
- The *(worst-case) communication complexity* of a distributed algorithm $\mathcal{DA}$ on a network $\mathcal{S}$, denoted by $\mathrm{CC}(\mathcal{DA}, \mathcal{S})$, is the maximum number of basic messages transmitted over the entire network during the execution of $\mathcal{DA}$ among all allowable initial states until termination.

Remark 3.4 (Space complexity conventions) *By convention, each processor knows its identity, i.e., it requires* $\log(n)$ *bits to represent its unique identifier in a set with* n *distinct elements. We do not count this cost in the space complexity of an algorithm.* •

We conclude this section by discussing ways of quantifying time, space, and communication complexity. The idea, borrowed from combinatorial optimization, is to adopt asymptotic "order of magnitude" measures. Formally, complexity bounds will be expressed with respect to the Bachman-Laundau symbols O, Ω, and Θ. Let us be more specific. In the following definitions, f denotes a function from $\mathbb{N}$ to $\mathbb{R}$.

(i) We say that an algorithm has time complexity *of order* $\Omega(f(n))$ *over some network* if, for all n, there exists a network of order n and initial processor values such that the time complexity of the algorithm is greater than a constant factor times $f(n)$.

(ii) We say that an algorithm has time complexity *of order* $O(f(n))$ *over arbitrary networks* if, for all n, for all networks of order n and for all initial processor values the time complexity of the algorithm is lower than a constant factor times $f(n)$.

(iii) We say that an algorithm has time complexity *of order* $\Theta(f(n))$ if its time complexity is of order $\Omega(f(n))$ over some network and $O(f(n))$ over arbitrary networks at the same time.

We use similar conventions for space and communication complexity.

In many cases the complexity of an algorithm will typically depend upon the number of nodes of the network. It is therefore useful to present a few simple facts about these functions now. Over arbitrary digraphs $\mathcal{S} = (I, E_{\mathrm{cmm}})$ of order n, we have

$$\mathrm{diam}(\mathcal{S}) \in \Theta(n), \quad |E_{\mathrm{cmm}}(\mathcal{S})| \in \Theta(n^2) \quad \text{and} \quad \mathrm{radius}(v, \mathcal{S}) \in \Theta(\mathrm{diam}(\mathcal{S})),$$

where v is any node of $\mathcal{S}$.

Remark 3.5 *Numerous variations of these definitions are possible. Even though we will not pursue them here, let us provide some pointers.*

(i) In the definition of lower bound, consider the logic quantifier describing the role of the network. The lower bound statement is "existential" rather than "global," in the sense that the bound does not hold for all graphs. As discussed in [2], it is possible to define also "global" lower bounds, i.e., lower bounds over all graphs, or lower bounds over specified classes of graphs.

(ii) The complexity notions introduced above focus on the worst-case situation. It is also possible to define expected *or* average *complexity notions, where one might be interested in characterizing, for example, the average number of rounds required or the average number of basic messages transmitted over the entire network during the execution of an algorithm among all allowable initial states until termination.*

(iii) It is possible to define complexity notions for problems, rather than algorithms, by considering, for example, the worst-case optimal performance among all algorithms that solve the given problem, or over classes of algorithms or classes of graphs. •

3.3 Leader Election

We formulate here a classical problem in distributed networks and summarize its complexity measures.

Problem 3.6 (Leader election) *Assume that all processors of a network have a state variable, say* `leader`*, initially set to* `unknown`*. We say that a leader is elected when one and only one processor has the state variable set to* `true` *and all others have it set to* `false`*. Elect a leader.* •

This is a task that is a bit more global in nature. We display here a solution that requires individual processors to know the diameter of the network, denoted by $\mathrm{diam}(\mathcal{S})$, or an upper bound on it.

> *[Informal description]* At each communication round, each agent sends to all its neighbors the maximum UID it has received up to that time. This is repeated for $\mathrm{diam}(\mathcal{S})$ rounds. At the last round, each agent compares the maximum received UID with its own, and declares itself a leader if they coincide, or a non-leader otherwise.

The algorithm is called FLOODMAX: the maximum UID in the network is transmitted to other agents in an incremental fashion. At the first communication round, agents that are neighbors of the agent with the maximum UID receive the message from it. At the next communication round, the neighbors of these

agents receive the message with the maximum UID. This process goes on for $\mathrm{diam}(\mathcal{S})$ rounds to ensure that every agent receives the maximum UID. Note that there are networks for which all agents receive the message with the maximum UID in fewer communication rounds than $\mathrm{diam}(\mathcal{S})$. The algorithm is formally stated as follows.

`Synchronous Network:` $\mathcal{S} = (\{1, \ldots, n\}, E_{\mathrm{cmm}})$

`Distributed Algorithm:` FLOODMAX

`Alphabet:` $\mathbb{A} = \{1, \ldots, n\} \cup \{\texttt{null}\}$

`Processor State:` $w = (\texttt{my-id}, \texttt{max-id}, \texttt{leader}, \texttt{round})$, where

`my-id`	$\in \{1, \ldots, n\}$,	initially:	$\texttt{my-id}^{[i]} = i$ for all i
`max-id`	$\in \{1, \ldots, n\}$,	initially:	$\texttt{max-id}^{[i]} = i$ for all i
`leader`	$\in \{\texttt{false}, \texttt{true}, \texttt{unknown}\}$,	initially:	$\texttt{leader}^{[i]} = \texttt{unknown}$ for all i
`round`	$\in \{0, 1, \ldots, \mathrm{diam}(\mathcal{S})\}$,	initially:	$\texttt{round}^{[i]} = 0$ for all i

`function` $\mathrm{msg}(w, i)$

1: **if** `round` $< \mathrm{diam}(\mathcal{S})$ **then**
2: **return** `max-id`
3: **else**
4: **return** `null`

`function` $\mathrm{stf}(w, y)$

1: `new-id`:= $\max\{$`max-id`, largest identifier in $y\}$
2: **case**
3: `round` $< \mathrm{diam}(\mathcal{S})$: `new-lead` := `unknown`
4: `round` $= \mathrm{diam}(\mathcal{S})$ AND `max-id` $=$ `my-id`: `new-lead` := `true`
5: `round` $= \mathrm{diam}(\mathcal{S})$ AND `max-id` $>$ `my-id`: `new-lead` := `false`
6: **return** (`my-id`, `new-id`, `new-lead`, `round` $+1$)

Figure 2 shows an execution of the FLOODMAX algorithm.

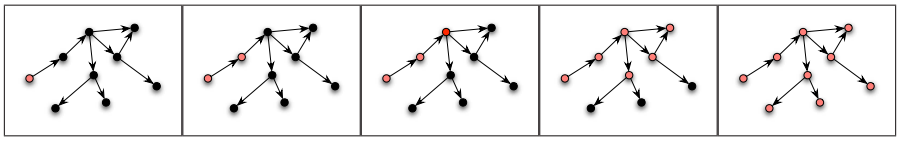


Figure 2: Execution of the FLOODMAX algorithm. The diameter of the network is 4. In the leftmost frame, the agent with the maximum UID is colored in red. After 4 communication rounds, its message has been received by all agents.

The properties of the algorithm are characterized in the following lemma. A complete analysis of this algorithm, including modifications to improve the communication complexity, is discussed in [1, Section 4.1].

Lemma 3.7 (Complexity upper bounds for the floodmax algorithm) *For a network $\mathcal{S}$ containing a spanning tree, the* FLOODMAX *algorithm has communication complexity in $O(\mathrm{diam}(\mathcal{S})|E_{\mathrm{cmm}}|)$, time complexity equal to $\mathrm{diam}(\mathcal{S})$, and space complexity in $\Theta(1)$.*

A simplification of the FLOODMAX algorithm leads to the Le Lann-Chang-Roberts (LCR) algorithm for leader election in rings, see [1, Chapter 3.3], that we describe next[1]. The LCR algorithm runs on a ring digraph and does not require the agents to know the diameter of the network.

> *[Informal description]* At each communication round, if the agent receives from its in-neighbor a UID that is larger than the UIDs received earlier, then the agent records the received UID and forwards it to the out-neighbor during the following communication round. (Agents do not record the number of communication rounds.) When the agent with the maximum UID receives its own UID from a neighbor, it declares itself the leader.

The algorithm is formally stated as follows.

`Synchronous Network:` ring digraph

`Distributed Algorithm:` LCR

`Alphabet:` $\mathbb{A} = \{1, \ldots, n\} \cup \{\texttt{null}\}$

`Processor State:` $w = (\texttt{my-id}, \texttt{max-id}, \texttt{leader}, \texttt{snd-flag})$, where

`my-id`	$\in \{1, \ldots, n\}$,	initially:	$\texttt{my-id}^{[i]} = i$ for all i
`max-id`	$\in \{1, \ldots, n\}$,	initially:	$\texttt{max-id}^{[i]} = i$ for all i
`leader`	$\in \{\texttt{true}, \texttt{false}, \texttt{unknown}\}$,	initially:	$\texttt{leader}^{[i]} = \texttt{unknown}$ for all i
`snd-flag`	$\in \{\texttt{true}, \texttt{false}\}$,	initially:	$\texttt{snd-flag}^{[i]} = \texttt{true}$ for all i

`function` msg(w, i)

```
1: if snd-flag = true then
2:    return max-id
3: else
4:    return null
```

`function` stf(w, y)

```
1: case
2:   (y contains only null msgs) OR (largest identifier in y < my-id):
3:       new-id := max-id
4:       new-lead := leader
5:       new-snd-flag := false
6:   (largest identifier in y = my-id):
7:       new-id := max-id
```

[1]Note that the description of the LCR algorithm given here is slightly different from the classic one as presented in [1].

```
        new-lead := true
        new-snd-flag := false
    (largest identifier in y > my-id):
        new-id := largest identifier in y
        new-lead := false
        new-snd-flag := true
return (my-id, new-id, new-lead, new-snd-flag)
```

Figure 3 shows an execution of the LCR algorithm. The properties of the

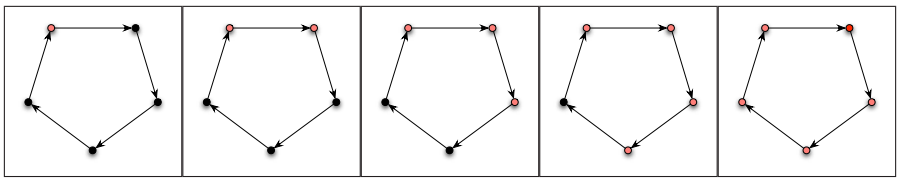


Figure 3: Execution of the LCR algorithm. In the leftmost frame, the agent with the maximum UID is colored in red. After 5 communication rounds, this agent receives its own UID from its in-neighbor and declares itself the leader.

LCR algorithm can be characterized as follows [1].

Lemma 3.8 (Complexity upper bounds for the LCR algorithm) *For a ring network $\mathcal{S}$ of order n, the* LCR *algorithm has communication complexity in $\Theta(n^2)$, time complexity equal to n, and space complexity in $\Theta(1)$.*

We conclude with a short remark. Strictly speaking, the LCR algorithm only achieves partially the leader election task as defined in Problem 3.6 because the agents other than the leader do not declare themselves as non-leaders. The algorithm can easily be modified to achieve fully the leader election task as follows: after the agent with the maximum UID has declared itself the leader, it can broadcast to the other agents the information that the leader has been elected.

3.4 Averaging Algorithms

This section provides a brief introduction to a special class of distributed algorithms called averaging algorithms. The synchronous version of averaging algorithms can be modeled within the framework of synchronous networks. In an averaging algorithm, each processor updates its state by computing a weighted linear combination of the state of its neighbors. Computing linear combinations of the initial states of the processors is one the most basic computations that a network can implement. Averaging algorithms find application in optimization, distributed decision-making, e.g., collective synchronization, and have a long and rich history, see e.g., [5, 6, 7, 8, 9, 10]. The richness comes from the vivid analogies with physical processes of diffusion, with Markov chain models,

and the theory of positive matrices developed by Perron and Frobenius, see e.g., [11, 12, 13].

Averaging algorithms are defined by stochastic matrices. For completeness, let us recall some basic linear algebra definitions. A matrix $A \in \mathbb{R}^{n\times n}$ with entries a_{ij}, $i, j \in \{1, \dots, n\}$, is

(i) *nonnegative* (resp., *positive*) if all its entries are nonnegative (resp., positive);

(ii) *row-stochastic* (or *stochastic* for brevity) if it is nonnegative and $\sum_{j=1}^{n} a_{ij} = 1$, for all $i \in \{1, \dots, n\}$; in other words, A is row-stochastic if

$$A\mathbf{1}_n = \mathbf{1}_n;$$

where $\mathbf{1}_n = (1, \dots, 1)^T \in \mathbb{R}^n$.

(iii) *doubly stochastic* if it is row-stochastic and column-stochastic, where we say that A is *column-stochastic* if $\mathbf{1}_n^T A = \mathbf{1}_n^T$;

(iv) *irreducible* if, for any nontrivial partition $J \cup K$ of the index set $\{1, \dots, n\}$, there exists $j \in J$ and $k \in K$ such that $a_{jk} \neq 0$.

We are now ready to introduce the class of averaging algorithms. The *averaging algorithm* associated to a sequence of stochastic matrices $\{F(\ell) \mid \ell \in \mathbb{Z}_{\geq 0}\} \subset \mathbb{R}^{n\times n}$ is the discrete-time dynamical system

$$w(\ell + 1) = F(\ell) \cdot w(\ell), \quad \ell \in \mathbb{Z}_{\geq 0}. \tag{1}$$

In the literature, averaging algorithms are also often referred to as agreement algorithms or as consensus algorithms.

Averaging algorithms are naturally associated with weighted digraphs, i.e., digraphs whose edges have weights. More precisely, a *weighted digraph* is a triplet $G = (V, E, A)$ where $V = \{v_1, \dots, v_n\}$ and E are a digraph and where $A \in \mathbb{R}_{\geq 0}^{n\times n}$ is a *weighted adjacency matrix* with the following properties: for $i, j \in \{1, \dots, n\}$, the entry $a_{ij} > 0$ if (v_i, v_j) is an edge of G, and $a_{ij} = 0$ otherwise. In other words, the scalars a_{ij}, for all $(v_i, v_j) \in E$, are a set of weights for the edges of G. Note that edge set is uniquely determined by the weighted adjacency matrix and it can be therefore omitted. The *weighted out-degree matrix* $D_{\text{out}}(G)$ and the *weighted in-degree matrix* $D_{\text{in}}(G)$ are the diagonal matrices defined by

$$D_{\text{out}}(G) = \text{diag}(A\mathbf{1}_n), \quad \text{and} \quad D_{\text{in}}(G) = \text{diag}(A^T\mathbf{1}_n).$$

The weighted digraph G is *weight-balanced* if $D_{\text{out}}(G) = D_{\text{in}}(G)$. Given a nonnegative $n \times n$ matrix A, its *associated weighted digraph* is the weighted digraph with nodes $\{1, \dots, n\}$, and weighted adjacency matrix A. The unweighted version of this weighted digraph is called the *associated digraph*. The following statements can be proven:

(i) if A is stochastic, then its associated digraph has weighted out-degree matrix equal to I_n;

(ii) if A is doubly stochastic, then its associated weighted digraph is weight-balanced and additionally both in-degree and out-degree matrices are equal to I_n;

(iii) A is irreducible if and only if its associated weighted digraph is strongly connected.

Next, we characterize the convergence properties of averaging algorithms. Let us introduce a useful property of collections of stochastic matrices. Given $\alpha \in]0,1]$, the set of *non-degenerate matrices with respect to* α consists of all stochastic matrices F with entries f_{ij}, for $i,j \in \{1,\dots,n\}$, satisfying

$$f_{ii} \in [\alpha, 1], \quad \text{and} \quad f_{ij} \in \{0\} \cup [\alpha, 1] \text{ for } j \neq i.$$

Additionally, the sequence of stochastic matrices $\{F(\ell) \mid \ell \in \mathbb{Z}_{\geq 0}\}$ is *non-degenerate* if these exists $\alpha \in]0,1]$ such that $F(\ell)$ is non-degenerate with respect to α for all $\ell \in \mathbb{Z}_{\geq 0}$.

We now state the following convergence result from [10].

Theorem 3.9 (Convergence for time-dependent stochastic matrices) *Let $\{F(\ell) \mid \ell \in \mathbb{Z}_{\geq 0}\} \subset \mathbb{R}^{n\times n}$ be a non-degenerate sequence of stochastic matrices. For $\ell \in \mathbb{Z}_{\geq 0}$, let $G(\ell)$ be the unweighted digraph associated to $F(\ell)$. The following statements are equivalent:*

(i) the set $\mathrm{diag}(\mathbb{R}^n)$ *is uniformly globally attractive for the associated averaging algorithm, that is, every evolution of the averaging algorithm at any time ℓ_0, approaches the set* $\mathrm{diag}(\mathbb{R}^n)$ *in the following* time-uniform *manner:*

> *for all $\ell_0 \in \mathbb{Z}_{\geq 0}$, for all $w_0 \in \mathbb{R}^n$, and for all neighborhoods W of* $\mathrm{diag}(\mathbb{R}^n)$*, there exists a single $\tau_0 \in \mathbb{Z}_{\geq 0}$ such that the evolution $w : [\ell_0, +\infty[\to \mathbb{R}^n$ defined by $w(\ell_0) = w_0$, takes value in W for all times $\ell \geq \ell_0 + \tau_0$.*

(ii) there exists a duration $\delta \in \mathbb{N}$ such that, for all $\ell \in \mathbb{Z}_{\geq 0}$, the digraph

$$G(\ell+1) \cup \cdots \cup G(\ell+\delta)$$

contains a globally reachable vertex.

4 Distributed Algorithms for Robotic Networks

This section describes models and algorithms for groups of robots that process information, sense, communicate, and move. We refer to such systems as robotic networks. In this section we review and survey a few modeling and algorithmic

topics in robotic coordination; earlier versions of this material were originally presented in [14, 15, 16, 17].

The section is organized as follows. First, we present the physical components of a network, that is, the mobile robots and the communication service connecting them. We then present the notion of control and communication law, and how a law is executed by a robotic network. We then discuss complexity notions for robotic networks. As an example of these notions, we introduce a simple law, called the agree-and-pursue law, which combines ideas from leader election algorithms and from cyclic pursuit (i.e., a game in which robots chase each other in a circular environment). We then consider in some detail algorithms for two basic motion coordination tasks, namely aggregation and deployment. We briefly formalize these problems and provide some basic algorithms for these two tasks.

4.1 Robotic Networks and Complexity

The global behavior of a robotic network arises from the combination of the local actions taken by its members. Each robot in the network can perform a few basic tasks such as sensing, communicating, processing information, and moving according to it. The many ways in which these capabilities can be integrated make a robotic network a versatile and, at the same time, complex system. The following robotic network model provides the framework to formalize, analyze, and compare distinct distributed behaviors.

We consider *uniform networks of robotic agents* defined by a tuple $\mathcal{S} = (I, \mathcal{R}, E_{\text{cmm}})$ consisting of a set of unique identifiers $I = \{1, \ldots, n\}$, a collection of control systems $\mathcal{R} = \{R^{[i]}\}_{i \in I}$, with $R^{[i]} = (X, U, X_0, f)$, and a map E_{cmm} from X^n to the subsets of $I \times I$ called the *communication edge map*. Here, (X, U, X_0, f) is a control system with state space $X \subset \mathbb{R}^d$, input space U, set of allowable initial states $X_0 \subset X$, and system dynamics $f : X \times U \to X$. An edge between two identifiers in E_{cmm} implies the ability of the corresponding two robots to exchange messages. A *control and communication law* for $\mathcal{S}$ consists of the sets:

(i) $\mathbb{A}$, called the *communication language*, whose elements are called *messages*;

(ii) W, set of values of some *processor variables* $w^{[i]} \in W$, $i \in I$, and $W_0 \subseteq W$, subset of *allowable initial values*. These sets correspond to the capability of robots to allocate additional variables and store sensor or communication data;

and the maps:

(iii) $\text{msg} : X \times W \times I \to \mathbb{A}$, called *message-generation function*;

(iv) $\text{stf} : X \times W \times \mathbb{A}^n \to W$, called *state-transition function*;

(v) $\text{ctl} : X \times W \times \mathbb{A}^n \to U$, called *control function*.

To implement a control and communication law each robot performs the following sequence or cycle of actions. At each instant $\ell \in \mathbb{Z}_{\geq 0}$, each robot i communicates to each robot j such that (i,j) belongs to $E_{\mathrm{cmm}}(x^{[1]},\dots,x^{[n]})$. Each robot i sends a message computed by applying the message-generation function to the current values of $x^{[i]}$ and $w^{[i]}$. After a negligible period of time, robot i resets the value of its logic variables $w^{[i]}$ by applying the state-transition function to the current value of $w^{[i]}$, and to the messages $y^{[i]}(\ell)$ received at ℓ. Between communication instants, i.e., for $t \in [\ell, \ell+1)$, robot i applies a control action computed by applying the control function to its state at the last sample time $x^{[i]}(\ell)$, the current value of $w^{[i]}$, and to the messages $y^{[i]}(\ell)$ received at ℓ.

Remark 4.1 (Algorithm properties and congestion models) *(i) In our present definition, all robots are identical and implement the same algorithm; in this sense the control and communication law is called* uniform *(or anonymous). If $W = W_0 = \emptyset$, then the control and communication law is* static *(or memoryless) and no state-transition function is defined. It is also possible for a law to be* time-independent *if the three relevant maps do not depend on time. Finally, let us also remark that this is a synchronous model in which all robots share a common clock.*

(ii) Communication and physical congestion affect the performance of robotic networks. These effects can be modeled by characterizing how the network parameters vary as the number of robots becomes larger. For example, in an ad hoc networks with n uniformly randomly placed nodes, it is known [18] that the maximum-throughput communication range $r(n)$ of each node decreases as the density of nodes increases; in d dimensions the appropriate scaling law is $r(n) \in \Theta\big((\log(n)/n)^{1/d}\big)$. As a second example, it is reasonable to assume that, as the number of robots increase, so should the area available for their motion. An alternative convenient approach is the one taken by [19], where robots' safety zones decrease with decreasing robots' speed. This suggests that, in a fixed environment, individual nodes of a large ensemble have to move at a speed decreasing with n, and in particular, at a speed proportional to $n^{-1/d}$. •

Next, we establish the notion of coordination task and of task achievement by a robotic network. Let $\mathcal{S}$ be a robotic network and let $\mathcal{W}$ be a set. A *coordination task* for $\mathcal{S}$ is a map $\mathcal{T} : X^n \times \mathcal{W}^n \to \{\texttt{true}, \texttt{false}\}$. If $\mathcal{W}$ is a singleton, then the coordination task is said to be *static* and can be described by a map $\mathcal{T}\colon X^n \to \{\texttt{true}, \texttt{false}\}$. Additionally, let $\mathcal{CC}$ a control and communication law for $\mathcal{S}$.

(i) The law $\mathcal{CC}$ is *compatible* with the task $\mathcal{T}\colon X^n \times \mathcal{W}^n \to \{\texttt{true}, \texttt{false}\}$ if its processor state take values in $\mathcal{W}$, that is, if $W^{[i]} = \mathcal{W}$, for all $i \in I$.

(ii) The law $\mathcal{CC}$ *achieves* the task $\mathcal{T}$ if it is compatible with it and if, for all initial conditions $x_0^{[i]} \in X_0$ and $w_0^{[i]} \in W_0$, $i \in I$, there exists $T \in \mathbb{Z}_{\geq 0}$ such that the network evolution $\ell \mapsto (x(\ell), w(\ell))$ has the property that $\mathcal{T}(x(\ell), w(\ell)) = \texttt{true}$ for all $\ell \geq T$.

In control-theoretic terms, achieving a task means establishing a convergence or stability result. Beside this key objective, one might be interested in efficiency as measured by required communication service, required control energy or by speed of completion. We focus on the latter notion.

(i) The *(worst-case) time complexity to achieve $\mathcal{T}$ with $\mathcal{CC}$ from* $(x_0, w_0) \in X_0^n \times W_0^n$ is

$$\mathrm{TC}(\mathcal{T}, \mathcal{CC}, x_0, w_0) = \inf \{\ell \mid \mathcal{T}(x(k), w(k)) = \texttt{true}\,, \text{ for all } k \geq \ell\}\,,$$

where $\ell \mapsto (x(\ell), w(\ell))$ is the evolution of $(\mathcal{S}, \mathcal{CC})$ from the initial condition (x_0, w_0);

(ii) The *(worst-case) time complexity to achieve $\mathcal{T}$ with $\mathcal{CC}$* is

$$\mathrm{TC}(\mathcal{T}, \mathcal{CC}) = \sup \Big\{ \mathrm{TC}(\mathcal{T}, \mathcal{CC}, x_0, w_0) \mid (x_0, w_0) \in X_0^n \times W_0^n \Big\}.$$

Some ideas on how to define meaningful notions of space and communication complexity are discussed in [14]. In the following discussion, we describe three coordination algorithms, which have been cast into this modeling framework and whose time complexity properties have been analyzed.

4.2 Agree-and-Pursue Algorithm

We begin our list of distributed algorithms with a simple law that is related to leader election algorithms, see Section 3.3, and to cyclic pursuit algorithms as studied in the control literature. Despite the apparent simplicity, this example is remarkable in that it combines a leader election task (in the processor states) with a uniform robotic deployment task (in the physical state), arguably two of the most basic tasks in distributed algorithms and cooperative control, respectively.

We consider n robots $\{\theta^{[1]}, \ldots, \theta^{[n]}\}$ in $\mathbb{S}^1$, moving along on the unit circle with angular velocity equal to the control input. Each robot is described by the tuple $(\mathbb{S}^1, [-u_{\max}, u_{\max}], \mathbb{S}^1, (0, \boldsymbol{e}))$, where $\boldsymbol{e}$ is the vector field on $\mathbb{S}^1$ describing unit-speed counterclockwise rotation. We assume that each robot can sense its own position and can communicate to any other robot within distance r along the circle. These data define the uniform robotic network $\mathcal{S}_{\mathrm{circle}}$.

[Algorithm description] The processor state consists of `dir` (the robot's direction of motion) taking values in $\{\texttt{c}, \texttt{cc}\}$ (meaning clockwise and counterclockwise) and `max-id` (the largest UID received by the robot, initially set to the robot's UID) taking values in I. At each communication round, each robot transmits its position and its processor state. Among the messages received from the robots moving towards its position, each robot picks the message with the largest value of `max-id`. If this value is larger than its own value,

the agent resets its processor state with the selected message. Between communication rounds, each robot moves in the clockwise or counterclockwise direction depending on whether its processor state `dir` is `c` or `cc`. Each robot moves k_{prop} times the distance to the immediately next neighbor in the chosen direction, or, if no neighbors are detected, k_{prop} times the communication range r.

For this network and this law there are two tasks of interest. First, we define the *direction agreement task* $\mathcal{T}_{\texttt{dir}} : (\mathbb{S}^1)^n \times W^n \to \{\texttt{true}, \texttt{false}\}$ by

$$\mathcal{T}_{\texttt{dir}}(\theta, w) = \begin{cases} \texttt{true}, & \text{if } \texttt{dir}^{[1]} = \cdots = \texttt{dir}^{[n]}, \\ \texttt{false}, & \text{otherwise}, \end{cases}$$

where $\theta = (\theta^{[1]}, \ldots, \theta^{[n]})$, $w = (w^{[1]}, \ldots, w^{[n]})$, and $w^{[i]} = (\texttt{dir}^{[i]}, \texttt{max-id}^{[i]})$, for $i \in I$. Furthermore, for $\varepsilon > 0$, we define the static *equidistance task* $\mathcal{T}_{\varepsilon\text{-eqdstnc}} : (\mathbb{S}^1)^n \to \{\texttt{true}, \texttt{false}\}$ to be `true` if and only if

$$\left| \min_{j \neq i} \text{dist}_{\texttt{c}}(\theta^{[i]}, \theta^{[j]}) - \min_{j \neq i} \text{dist}_{\texttt{cc}}(\theta^{[i]}, \theta^{[j]}) \right| < \varepsilon, \quad \text{for all } i \in I.$$

In other words, $\mathcal{T}_{\varepsilon\text{-eqdstnc}}$ is true when, for every agent, the distance to the closest clockwise neighbor and to the closest counterclockwise neighbor are approximately equal.

An implementation of this control and communication law is shown in Figure 4. As parameters we select $n = 45$, $r = 2\pi/40$, $u_{\max} = 1/4$ and $k_{\text{prop}} = 7/16$. Along the evolution, all robots agree upon a common direction of motion and, after suitable time, they reach a uniform distribution. A careful analysis based

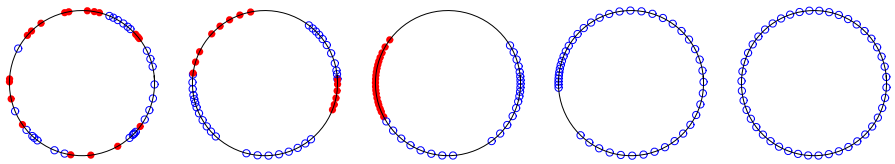


Figure 4: The AGREE & PURSUE law. Disks and circles correspond to robots moving counterclockwise and clockwise, respectively. The initial positions and the initial directions of motion are randomly generated. The five pictures depict the network state at times 0, 9, 20, 100, and 800.

on invariance properties and Lyapunov functions allows us to establish that, under appropriate conditions, indeed both tasks are achieved by the agree-and-pursue law [14].

Theorem 4.2 (Time complexity of agree-and-pursue law) *For $k_{\text{prop}} \in]0, \frac{1}{2}[$, in the limit as $n \to +\infty$ and $\varepsilon \to 0^+$, the network $\mathcal{S}_{\text{circle}}$ with $u_{\max}(n) \geq k_{\text{prop}} r(n)$, the law $\mathcal{CC}_{\text{AGREE \& PURSUE}}$, and the tasks $\mathcal{T}_{\texttt{dir}}$ and $\mathcal{T}_{\varepsilon\text{-eqdstnc}}$ together satisfy:*

(i) $\mathrm{TC}(\mathcal{T}_{\mathtt{dir}}, \mathcal{CC}_{\text{AGREE \& PURSUE}}) \in \Theta(r(n)^{-1})$;

(ii) *if $\delta(n)$ is lower bounded by a positive constant as $n \to +\infty$, then*

$$\mathrm{TC}(\mathcal{T}_{\varepsilon\text{-eqdstnc}}, \mathcal{CC}_{\text{AGREE \& PURSUE}}) \in \Omega(n^2 \log(n\varepsilon)^{-1}),$$
$$\mathrm{TC}(\mathcal{T}_{\varepsilon\text{-eqdstnc}}, \mathcal{CC}_{\text{AGREE \& PURSUE}}) \in O(n^2 \log(n\varepsilon^{-1})).$$

If $\delta(n)$ is upper bounded by a negative constant, then $\mathcal{CC}_{\text{AGREE \& PURSUE}}$ does not achieve $\mathcal{T}_{\varepsilon\text{-eqdstnc}}$ in general.

Finally we compare these results with the complexity result known for the leader election problem.

Remark 4.3 (Comparison with leader election) *Let us compare the agree-and-pursue control and communication law with the classical Le Lann-Chang-Roberts (LCR) algorithm for leader election discussed in Section 3.3. The leader election task consists of electing a unique agent among all robots in the network; it is therefore different from, but closely related to, the coordination task $\mathcal{T}_{\mathtt{dir}}$. The LCR algorithm operates on a static network with the ring communication topology, and achieves leader election with time and total communication complexity, respectively, $\Theta(n)$ and $\Theta(n^2)$. The agree-and-pursue law operates on a robotic network with the $r(n)$-disk communication topology, and achieves $\mathcal{T}_{\mathtt{dir}}$ with time and total communication complexity, respectively, $\Theta(r(n)^{-1})$ and $O(n^2 r(n)^{-1})$. If wireless communication congestion is modeled by $r(n)$ of order $1/n$ as in Remark 4.1, then the two algorithms have identical time complexity and the LCR algorithm has better communication complexity. Note that computations on a possibly disconnected, dynamic network are more complex than on a static ring topology.* •

4.3 Aggregation Algorithms

The rendezvous objective (also referred to as the gathering problem) is to achieve agreement over the location of the robots, that is, to steer each agent to a common location. An early reference on this problem is [20]; more recent references include [21, 22, 23, 24]. We consider two scenarios which differ in the robots' communication capabilities and the environment in which the robots move. First [23], we consider the problem of rendezvous for robots equipped with *range-limited communication* in obstacle-free environments. In this case, each robot is capable of sensing its position in the Euclidean space $\mathbb{R}^d$ and can communicate it to any other robot within a given distance r. This communication service is modeled by the r-disk graph, in which two robots are neighbors if and only if their Euclidean distance is less than or equal to r. Second [25], we consider *visually-guided robots*. Here the robots are assumed to move in a nonconvex simple polygonal environment Q. Each robot can sense, within line of sight, any other robot as well as the distance to the boundary of the environment. The relationship between the robots can be characterized by the so-called

visibility graph: two robots are neighbors if and only if they are mutually visible to each other.

In both scenarios, the rendezvous problem cannot be solved with distributed information unless the robots' initial positions form a connected communication graph. Arguably, a good property of any rendezvous algorithm is that of maintaining connectivity between robots. This connectivity-maintenance objective is interesting on its own. It turns out that this objective can be achieved through local constraints on the robots' motion. Motion constraint sets that maintain connectivity are designed in [20, 25] by exploiting the geometric properties of disk and visibility graphs.

These discussions lead to the following algorithm that solves the rendezvous problems for both communication scenarios. The robots execute what is known as the *Circumcenter Algorithm*; here is an informal description. Each robot iteratively performs the following tasks:

```
1: acquire neighbors' positions
2: compute connectivity constraint set
3: move toward the circumcenter of the point set comprised of its neighbors
   and of itself, while remaining inside the connectivity constraint
   set.
```

One can prove that, under technical conditions, the algorithm does achieve the rendezvous task in both scenarios; see [23, 25]. Additionally, when $d = 1$, it can be shown that the time complexity of this algorithm is $\Theta(n)$; see [15].

4.4 Deployment Algorithms

The problem of deploying a group of robots over a given region of interest can be tackled with the following simple heuristic. Each robot iteratively performs the following tasks:

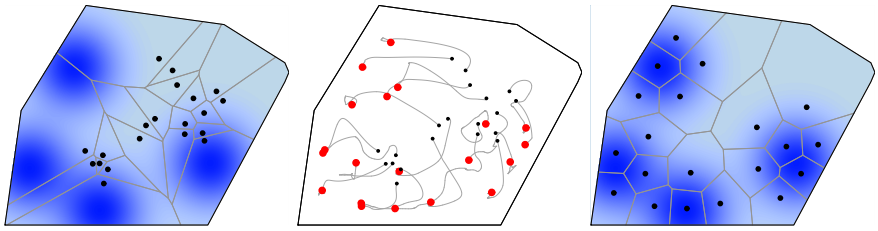
```
1: acquire neighbors' positions
2: compute own dominance region
3: move towards the center of own dominance region
```

This short description can be made accurate by specifying what notions of dominance region and of center are to be adopted. In what follows we mention two examples and refer to [26, 27, 28, 29] for more details.

First, we consider the *area-coverage deployment problem* in a convex environment Q. The objective is to maximize the area within close range of the mobile nodes. This models a scenario in which the nodes are equipped with some sensors that take measurements of some physical quantity in the environment, e.g., temperature or concentration. Assume that certain regions in the environment are more important than others and describe this by a density function ϕ. This problems leads to the coverage performance metric

$$\mathcal{H}_{\text{ave}}(p_1, \ldots, p_n) = \int_Q \min_{i \in \{1,\ldots,n\}} f(\|q - p_i\|)\, \phi(q) dq = \sum_{i=1}^{n} \int_{V_i} f(\|q - p_i\|) \phi(q) dq.$$

Here p_i is the position of the ith node, f measures the performance of an individual sensor, and $\{V_1, \ldots, V_n\}$ is the Voronoi partition of the environment Q generated by the positions $\{p_1, \ldots, p_n\}$. If we assume that each node obeys a first-order dynamical behavior, then a simple gradient scheme can be easily implemented in a spatially-distributed manner. Following the gradient of $\mathcal{H}_{\text{ave}}$ corresponds, in the algorithm described above, to defining (1) the dominance regions to be the Voronoi cells generated by the robots, and (2) the center of a region to be the centroid of the region (if $f(x) = x^2$). Because the closed-loop system is a gradient flow for the cost function, performance is locally, continuously optimized. As a special case, when the environment is a segment and $\phi = 1$, the time complexity of the algorithm can be shown to be $O(n^3 \log(n\varepsilon^{-1}))$, where ε is an accuracy threshold below which we consider the task accomplished.



(a) (b) (c)

Figure 5: Deployment algorithm for the area-coverage problem. Each of the 20 robots moves toward the centroid of its Voronoi cell. This strategy corresponds to the network following the gradient of $\mathcal{H}_{\text{ave}}$. Areas of the polygon with greater importance are colored darker. Figures (a) and (c) show, respectively, the initial and final locations, with the corresponding Voronoi partitions. Figure (b) illustrates the gradient descent flow.

Second, we consider the problem of deploying *to maximize the likelihood of detecting a source*. For example, consider devices equipped with acoustic sensors attempting to detect a sound-source (or, similarly, antennas detecting RF signals, or chemical sensors localizing a pollutant source). For a variety of criteria, when the source emits a known signal and the noise is Gaussian, we know that the optimal detection algorithm involves a matched filter, that detection performance is a function of signal-to-noise-ratio, and, in turn, that signal-to-noise ratio is inversely proportional to the sensor-source distance. In this case, the appropriate cost function is

$$\mathcal{H}_{\text{worst}}(p_1, \ldots, p_n) = \max_{q \in Q} \min_{i \in \{1,\ldots,n\}} f(\|q - p_i\|) = \max_{q \in V_i} f(\|q - p_i\|),$$

and a greedy motion coordination algorithm is for each node to move toward the circumcenter of its Voronoi cell. A detailed analysis [28] shows that the detection likelihood is inversely proportional to the circumradius of each node's Voronoi cell, and that, if the nodes follow the algorithm described above, then the detection likelihood increases monotonically as a function of time.

5 Bibliographical Notes

In this section we present a necessarily incomplete discussion of some relevant literature that we have not yet mentioned in the previous sections.

First, we review some literature on emergent and self-organized swarming behaviors in biological groups with distributed agent-to-agent interactions. Interesting dynamical systems arise in biological networks at multiple levels of resolution, all the way from interactions among molecules and cells [MB01] to the behavioral ecology of animal groups [Oku86]. Flocks of birds and schools of fish can travel in formation and act as one unit (see [PVG02]), allowing these animals to defend themselves against predators and protect their territories. Wildebeest and other animals exhibit complex collective behaviors when migrating, such as obstacle avoiding, leader election, and formation keeping (see [Sin77, GL93]). Certain foraging behaviors include individual animals partitioning their environment into nonoverlapping zones (see [Bar74]). Honey bees [SB99], gorillas [SH94], and whitefaced capuchins [BC95] exhibit synchronized group activities such as initiation of motion and change of travel direction. These remarkable dynamic capabilities are achieved apparently without following a group leader; see [Oku86, PVG02, GL93, Bar74, SB99, SH94, BC95] for specific examples of animal species and [CKFL05, CR03] for general studies.

With regards to distributed motion coordination algorithms, much progress has been made on pattern formation [SY99, BK04, JK04, SPL07], flocking [TJP07, OS06], self-assembly [KGL06], swarm aggregation [GP03], gradient climbing [OFL04], cyclic pursuit [BCE91, MBF04, SBF05], vehicle routing [LH97, SSFV05], and connectivity maintenance problems [ZP05, SNB07].

Much research has been devoted to distributed task allocation problems. The work in [GM04] proposes a taxonomy of task allocation problems. In papers such as [GSH06, AH06, SCRW03, MP07], advanced heuristic methods are developed, and their effectiveness is demonstrated through simulation or real world implementation. Distributed auction algorithms are discussed in [CW03, MP07] building on the classic works in [BC91, BC93]. A distributed MILP solver is proposed in [AH06]. A spatially distributed receding-horizon scheme is proposed in [FB04, PFB07]. There has also been prior work on target assignment problems [AMS07, ZP07, SB07]. Target allocation for vehicles with nonholonomic constraints is studied in [RSD07, SFB08, SBF07].

References with a focus on robotic networks include the survey in [CFK97], the text [Ark98] on behavior-based robotics, and the recent special issue [APP02] of the IEEE Transaction on Robotics and Automation. An important contribution towards a network model of mobile interacting robots is introduced in [SY99]. This model consists of a group of identical "distributed anonymous mobile robots" characterized as follows: no explicit communication takes place between them, and at each time instant of an "activation schedule," each robot senses the relative position of all other robots and moves according to a prespecified algorithm. Communication complexity for control and communication algorithms A related model is presented in [24], where as few capabilities as possible are assumed on the agents, with the objective of understanding the

limitations of multi-agent networks. A brief survey of models, algorithms, and the need for appropriate complexity notions is presented in [San01]. Recently, a notion of communication complexity for control and communication algorithms in multi-robot systems is analyzed in [Kla03], see also [KM04].

Finally, with regards to linear distributed algorithms we mention the following references, on top of the ones discussed in Section 3.4. Various results are available on continuous-time consensus algorithms [OSM04, Mor04, RB05, LFM05, LFM07, FYL06], consensus over random networks [HM05, Wu06, TSJ07], consensus algorithms for general functions [BGP06, Cor08], connections with the heat equation and partial difference equation [FTBG06], convergence in time-delayed and asynchronous settings [BHOT05, AB06], quantized consensus problems [Sav04, KBS07], applications to distributed signal processing [SOSM05, XBL05, OSFFS06], characterization of the convergence rates and time complexity [LO81, OT07, CFSZ08, CMA08]. Finally, two recent surveys are [OSFM07, RBA07].

6 Future Directions

Robotic networks incorporate numerous subsystems. Their design is challenging because they integrate heterogeneous hardware and software components. Additionally, the operation of robotic networks is subject to computing, energy, cost, and safety constraints. The interaction with the physical world and the uncertain response from other members of the network are also integral parts to consider in the management of robotic networks. Traditional centralized approaches are not valid to satisfy the scalability requirements of these systems. Thus, in order to successfully deploy robotic and communication networks, it is necessary to expand our present knowledge about how to integrate and efficiently control them. The present chapter has offered a glimpse into these problems. We have presented verification and complexity tools to evaluate the cost of cooperative strategies that achieve a variety of tasks.

Networks of robotic agents are an example of the class of complex, networked systems which pervades our world. Understanding how to design robotic swarms requires the development of new fundamental theories that can explain the behavior of general networks evolving with time. Some of the desired properties of these systems are robustness, ease of control, predictability with time, guaranteed performance, and quality of service. Self-organization and distributed management would also allow for the minimal supervision necessary for scalability. However, devising systems that meet all these criteria is not an easy task. A small deviation by an agent or a change in certain parameters may produce a dramatic change in the overall network behavior. Relevant questions that pertain these aspects are currently being approached from a range of disciplines such as systems and control, operations research, random graph theory, statistical physics, and game theory over networks. Future work will undoubtedly lead to a cross-fertilization of these and other areas that will help design efficient robotic sensor networks.

Acknowledgments

This material is based upon work supported in part by NSF CAREER Award CMS-0643679, NSF CAREER Award ECS-0546871, and AFOSR MURI Award FA9550-07-1-0528.

Bibliography

[1] N. A. Lynch, *Distributed Algorithms*. San Mateo, CA: Morgan Kaufmann Publishers, 1997.

[2] D. Peleg, *Distributed Computing. A Locality-Sensitive Approach*, ser. Monographs on Discrete Mathematics and Applications. Philadelphia, PA: SIAM, 2000.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.

[4] R. G. Gallager, *Information Theory and Reliable Communication*. New York: John Wiley, 1968.

[5] M. H. DeGroot, "Reaching a consensus," *Journal of the American Statistical Association*, vol. 69, no. 345, pp. 118–121, 1974.

[6] G. Cybenko, "Dynamic load balancing for distributed memory multiprocessors," *Journal of Parallel and Distributed Computing*, vol. 7, pp. 279–301, 1989.

[7] J. N. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, MIT, Nov. 1984, Technical Report LIDS-TH-1424, Laboratory for Information and Decision Systems.

[8] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

[9] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, 2003.

[10] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[11] S. Chatterjee and E. Seneta, "Towards consensus: Some convergence theorems on repeated averaging," *Journal of Applied Probability*, vol. 14, no. 1, pp. 89–97, 1977.

[12] R. Cogburn, "The ergodic theory of Markov chains in random environments," *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, vol. 66, no. 1, 1984.

[13] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1985.

[14] S. Martínez, F. Bullo, J. Cortés, and E. Frazzoli, "On synchronous robotic networks – Part I: Models, tasks and complexity," *IEEE Transactions on Automatic Control*, vol. 52, no. 12, pp. 2199–2213, 2007.

[15] ——, "On synchronous robotic networks – Part II: Time complexity of rendezvous and deployment algorithms," *IEEE Transactions on Automatic Control*, vol. 52, no. 12, pp. 2214–2226, 2007.

[16] F. Bullo, "Notes on multi-agent motion coordination: Models and algorithms," in *Network Embedded Sensing and Control. (Proceedings of NESC'05 Worskhop)*, ser. Lecture Notes in Control and Information Sciences, P. J. Antsaklis and P. Tabuada, Eds. New York: Springer Verlag, 2006, vol. 331, pp. 3–8.

[17] S. Martínez, J. Cortés, and F. Bullo, "Motion coordination with distributed information," *IEEE Control Systems Magazine*, vol. 27, no. 4, pp. 75–88, 2007.

[18] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[19] V. Sharma, M. Savchenko, E. Frazzoli, and P. Voulgaris, "Transfer time complexity of conflict-free vehicle routing with no communications," *International Journal of Robotics Research*, vol. 26, no. 3, pp. 255–272, 2007.

[20] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita, "Distributed memoryless point convergence algorithm for mobile robots with limited visibility," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 818–828, 1999.

[21] J. Lin, A. S. Morse, and B. D. O. Anderson, "The multi-agent rendezvous problem: An extended summary," in *Proceedings of the 2003 Block Island Workshop on Cooperative Control*, ser. Lecture Notes in Control and Information Sciences, V. Kumar, N. E. Leonard, and A. S. Morse, Eds. New York: Springer Verlag, 2004, vol. 309, pp. 257–282.

[22] Z. Lin, M. Broucke, and B. Francis, "Local control strategies for groups of mobile autonomous agents," *IEEE Transactions on Automatic Control*, vol. 49, no. 4, pp. 622–629, 2004.

[23] J. Cortés, S. Martínez, and F. Bullo, "Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions," *IEEE Transactions on Automatic Control*, vol. 51, no. 8, pp. 1289–1298, 2006.

[24] P. Flocchini, G. Prencipe, N. Santoro, and P. Widmayer, "Gathering of asynchronous oblivious robots with limited visibility," *Theoretical Computer Science*, vol. 337, no. 1-3, pp. 147–168, 2005.

[25] A. Ganguli, J. Cortés, and F. Bullo, "On rendezvous for visually-guided agents in a nonconvex polygon," in *IEEE Conf. on Decision and Control and European Control Conference*, Seville, Spain, Dec. 2005, pp. 5686–5691.

[26] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[27] J. Cortés, S. Martínez, and F. Bullo, "Spatially-distributed coverage optimization and control with limited-range interactions," *ESAIM. Control, Optimisation & Calculus of Variations*, vol. 11, pp. 691–719, 2005.

[28] J. Cortés and F. Bullo, "Coordination and geometric optimization via distributed dynamical systems," *SIAM Journal on Control and Optimization*, vol. 44, no. 5, pp. 1543–1574, 2005.

[29] A. Ganguli, J. Cortés, and F. Bullo, "Distributed deployment of asynchronous guards in art galleries," in *American Control Conference*, Minneapolis, MN, June 2006, pp. 1416–1421.

References from Bibliographical Notes

[AB06] D. Angeli and P.-A. Bliman. Stability of leaderless discrete-time multi-agent systems. *Mathematics of Control, Signals and Systems*, 18(4):293–322, 2006.

[AH06] M. Alighanbari and J. P. How. Robust decentralized task assignment for cooperative UAVs. In *AIAA Conf. on Guidance, Navigation and Control*, Keystone, CO, August 2006.

[AMS07] G. Arslan, J. R. Marden, and J. S. Shamma. Autonomous vehicle-target assignment: A game theoretic formulation. *ASME Journal on Dynamic Systems, Measurement, and Control*, 129(5):584–596, 2007.

[APP02] T. Arai, E. Pagello, and L. E. Parker. Guest editorial: Advances in multirobot systems. *IEEE Transactions on Robotics and Automation*, 18(5):655–661, 2002.

[Ark98] R. C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, MA, 1998.

[Bar74] G. W. Barlow. Hexagonal territories. *Animal Behavior*, 22:876–878, 1974.

[BC91] D. P. Bertsekas and D. A. Castañón. Parallel synchronous and asynchronous implementations of the auction algorithm. *Parallel Computing*, 17:707–732, 1991.

[BC93] D. P. Bertsekas and D. A. Castañón. Parallel primal-dual methods for the minimum cost flow problem. *Computational Optimization and Applications*, 2(4):317–336, 1993.

[BC95] S. Boinski and A. F. Campbell. Use of trill vocalizations to coordinate troop movement among whitefaced capuchins - a 2nd field-test. *Behaviour*, 132:875–901, 1995.

[BCE91] A. M. Bruckstein, N. Cohen, and A. Efrat. Ants, crickets, and frogs in cyclic pursuit. Technical Report CIS 9105, Center for Intelligent Systems, Technion, Haifa, Israel, July 1991. Available electronically at `http://www.cs.technion.ac.il/tech-reports`.

[BGP06] D. Bauso, L. Giarré, and R. Pesenti. Nonlinear protocols for optimal distributed consensus in networks of dynamic agents. *Systems & Control Letters*, 55(11):918–928, 2006.

[BHOT05] V. D. Blondel, J. M. Hendrickx, A. Olshevsky, and J. N. Tsitsiklis. Convergence in multiagent coordination, consensus, and flocking. In *IEEE Conf. on Decision and Control and European Control Conference*, pages 2996–3000, Seville, Spain, December 2005.

[BK04] C. Belta and V. Kumar. Abstraction and control for groups of robots. *IEEE Transactions on Robotics*, 20(5):865–875, 2004.

[CFK97] Y. Uny Cao, A. S. Fukunaga, and A. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27, 1997.

[CFSZ08] R. Carli, F. Fagnani, A. Speranzon, and S. Zampieri. Communication constraints in the average consensus problem. *Automatica*, 44(3):671–684, 2008.

[CKFL05] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin. Effective leadership and decision-making in animal groups on the move. *Nature*, 433:513–516, 2005.

[CMA08] M. Cao, A. S. Morse, and B. D. O. Anderson. Reaching a consensus in a dynamically changing environment - convergence rates, measurement delays and asynchronous events. *SIAM Journal on Control and Optimization*, 47(2):601–623, 2008.

[Cor08] J. Cortés. Distributed algorithms for reaching consensus on general functions. *Automatica*, 44(3):726–737, 2008.

[CR03] L. Conradt and T. J. Roper. Group decision-making in animals. *Nature*, 421(6919):155–158, 2003.

[CW03] D. A. Castañón and C. Wu. Distributed algorithms for dynamic reassignment. In *IEEE Conf. on Decision and Control*, pages 13–18, Maui, HI, December 2003.

[FB04] E. Frazzoli and F. Bullo. Decentralized algorithms for vehicle routing in a stochastic time-varying environment. In *IEEE Conf. on Decision and Control*, pages 3357–3363, Paradise Island, Bahamas, December 2004.

[FTBG06] G. Ferrari-Trecate, A. Buffa, and M. Gati. Analysis of coordination in multi-agent systems through partial difference equations. *IEEE Transactions on Automatic Control*, 51(6):1058–1063, 2006.

[FYL06] R. A. Freeman, P. Yang, and K. M. Lynch. Stability and convergence properties of dynamic average consensus estimators. In *IEEE Conf. on Decision and Control*, pages 398–403, San Diego, CA, December 2006.

[GL93] S. Gueron and S. A. Levin. Self-organization of front patterns in large wildebeest herds. *Journal of Theoretical Biology*, 165:541–552, 1993.

[GM04] B. P. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939–954, 2004.

[GP03] V. Gazi and K. M. Passino. Stability analysis of swarms. *IEEE Transactions on Automatic Control*, 48(4):692–697, 2003.

[GSH06] M. F. Godwin, S. Spry, and J. K. Hedrick. Distributed collaboration with limited communication using mission state estimates. In *American Control Conference*, pages 2040–2046, Minneapolis, MN, June 2006.

[HM05] Y. Hatano and M. Mesbahi. Agreement over random networks. *IEEE Transactions on Automatic Control*, 50:1867–1872, 2005.

[JK04] E. W. Justh and P. S. Krishnaprasad. Equilibria and steering laws for planar formations. *Systems & Control Letters*, 52(1):25–38, 2004.

[KBS07] A. Kashyap, T. Başar, and R. Srikant. Quantized consensus. *Automatica*, 43(7):1192–1203, 2007.

[KGL06] E. Klavins, R. Ghrist, and D. Lipsky. A grammatical approach to self-organizing robotic systems. *IEEE Transactions on Automatic Control*, 51(6):949–962, 2006.

[Kla03] E. Klavins. Communication complexity of multi-robot systems. In J.-D. Boissonnat, J. W. Burdick, K. Goldberg, and S. Hutchinson, editors, *Algorithmic Foundations of Robotics V*, volume 7 of *Springer Tracts in Advanced Robotics*, Berlin Heidelberg, 2003. Springer Verlag.

[KM04] E. Klavins and R. M. Murray. Distributed algorithms for cooperative control. *IEEE Pervasive Computing*, 3(1):56–65, 2004.

[LFM05] Z. Lin, B. Francis, and M. Maggiore. Necessary and sufficient graphical conditions for formation control of unicycles. *IEEE Transactions on Automatic Control*, 50(1):121–127, 2005.

[LFM07] Z. Lin, B. Francis, and M. Maggiore. State agreement for continuous-time coupled nonlinear systems. *SIAM Journal on Control and Optimization*, 46(1):288–307, 2007.

[LH97] V. J. Lumelsky and K. R. Harinarayan. Decentralized motion planning for multiple mobile robots: the cocktail party model. *Autonomous Robots*, 4(1):121–135, 1997.

[LO81] H. J. Landau and A. M. Odlyzko. Bounds for eigenvalues of certain stochastic matrices. *Linear Algebra and its Applications*, 38:5–15, 1981.

[MB01] M. B. Miller and B. L. Bassler. Quorum sensing in bacteria. *Annual Review of Microbiology*, 55:165–199, 2001.

[MBF04] J. A. Marshall, M. E. Broucke, and B. A. Francis. Formations of vehicles in cyclic pursuit. *IEEE Transactions on Automatic Control*, 49(11):1963–1974, 2004.

[Mor04] L. Moreau. Stability of continuous-time distributed consensus algorithms, 2004. Preprint. Available electronically at `http://xxx.arxiv.org/math.OC/0409010`.

[MP07] B. J. Moore and K. M. Passino. Distributed task assignment for mobile agents. *IEEE Transactions on Automatic Control*, 52(4):749–753, 2007.

[OFL04] P. Ögren, E. Fiorelli, and N. E. Leonard. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *IEEE Transactions on Automatic Control*, 49(8):1292–1302, 2004.

[Oku86] A. Okubo. Dynamical aspects of animal grouping: swarms, schools, flocks and herds. *Advances in Biophysics*, 22:1–94, 1986.

[OS06] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, 2006.

[OSFFS06] R. Olfati-Saber, E. Franco, E. Frazzoli, and J. S. Shamma. Belief consensus and distributed hypothesis testing in sensor networks. In P.J. Antsaklis and P. Tabuada, editors, *Network Embedded Sensing and Control. (Proceedings of NESC'05 Worskhop)*, volume 331 of *Lecture Notes in Control and Information Sciences*, pages 169–182. Springer Verlag, New York, 2006.

[OSFM07] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *IEEE Proceedings*, 95(1):215–233, 2007.

[OSM04] R. Olfati-Saber and R. M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004.

[OT07] A. Olshevsky and J. N. Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM Journal on Control and Optimization*, 2007. Submitted.

[PFB07] M. Pavone, E. Frazzoli, and F. Bullo. Decentralized algorithms for stochastic and dynamic vehicle routing with general target distribution. In *IEEE Conf. on Decision and Control*, pages 4869–4874, New Orleans, LA, December 2007.

[PVG02] J. K. Parrish, S. V. Viscido, and D. Grunbaum. Self-organized fish schools: an examination of emergent properties. *Biological Bulletin*, 202:296–305, 2002.

[RB05] W. Ren and R. W. Beard. Consensus seeking in multi-agent systems under dynamically changing interaction topologies. *IEEE Transactions on Automatic Control*, 50(5):655–661, 2005.

[RBA07] W. Ren, R. W. Beard, and E. M. Atkins. Information consensus in multivehicle cooperative control: Collective group behavior through local interaction. *IEEE Control Systems Magazine*, 27(2):71–82, 2007.

[RSD07] S. Rathinam, R. Sengupta, and S. Darbha. A resource allocation algorithm for multi-vehicle systems with non holonomic constraints. *IEEE Transactions on Automation Sciences and Engineering*, 4(1):98–104, 2007.

[San01] N. Santoro. Distributed computations by autonomous mobile robots. In L. Pacholski and P. Ruzicka, editors, *SOFSEM 2001: Conference on Current Trends in Theory and Practice of Informatics (Piestany, Slovak Republic)*, volume 2234 of *Lecture Notes in Computer Science*, pages 110–115. Springer Verlag, New York, 2001.

[Sav04] A.V. Savkin. Coordinated collective motion of groups of autonomous mobile robots: Analysis of Vicsek's model. *IEEE Transactions on Automatic Control*, 49(6):981–982, 2004.

[SB99] T. D. Seeley and S. C. Buhrman. Group decision-making in swarms of honey bees. *Behavioral Ecology and Sociobiology*, 45:19–31, 1999.

[SB07] S. L. Smith and F. Bullo. Monotonic target assignment for robotic networks. *IEEE Transactions on Automatic Control*, June 2007. Submitted.

[SBF05] S. L. Smith, M. E. Broucke, and B. A. Francis. A hierarchical cyclic pursuit scheme for vehicle networks. *Automatica*, 41(6):1045–1053, 2005.

[SBF07] K. Savla, F. Bullo, and E. Frazzoli. Traveling Salesperson Problems for a double integrator. *IEEE Transactions on Automatic Control*, 2007. (Submitted Nov 2006) To appear.

[SCRW03] C. Schumacher, P. R. Chandler, S. J. Rasmussen, and D. Walker. Task allocation for wide area search munitions with variable path length. In *American Control Conference*, pages 3472–3477, Denver, CO, 2003.

[SFB08] K. Savla, E. Frazzoli, and F. Bullo. Traveling Salesperson Problems for the Dubins vehicle. *IEEE Transactions on Automatic Control*, 53(9), 2008. (Submitted Jun 2006) To appear.

[SH94] K. J. Stewart and A. H. Harcourt. Gorillas vocalizations during rest periods - signals of impending departure. *Behaviour*, 130:29–40, 1994.

[Sin77] A. R. Sinclair. *The African Buffalo, A Study of Resource Limitation of Population*. The University of Chicago Press, Chicago, IL, 1977.

[SNB07] K. Savla, G. Notarstefano, and F. Bullo. Maintaining limited-range connectivity among second-order agents. *SIAM Journal on Control and Optimization*, 2007. To appear, (Submitted Nov 2006).

[SOSM05] D. P. Spanos, R. Olfati-Saber, and R. M. Murray. Approximate distributed Kalman filtering in sensor networks with quantifiable performance. In *Symposium on Information Processing of Sensor Networks (IPSN)*, pages 133–139, Los Angeles, CA, April 2005.

[SPL07] R. Sepulchre, D. Paley, and N. E. Leonard. Stabilization of planar collective motion: All-to-all communication. *IEEE Transactions on Automatic Control*, 52:811–824, 2007.

[SSFV05] V. Sharma, M. Savchenko, E. Frazzoli, and P. Voulgaris. Time complexity of sensor-based vehicle routing. In S. Thrun, G. Sukhatme, S. Schaal, and O. Brock, editors, *Robotics: Science and Systems*, pages 297–304. MIT Press, Cambridge, MA, 2005.

[SY99] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.

[TJP07] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. Flocking in fixed and switching networks. *IEEE Transactions on Automatic Control*, 52(5):863–868, 2007.

[TSJ07] A. Tahbaz-Salehi and A. Jadbabaie. Consensus over random networks. *IEEE Transactions on Automatic Control*, 2007. To appear.

[Wu06] C. W. Wu. Synchronization and convergence of linear dynamics in random directed networks. *IEEE Transactions on Automatic Control*, 51(7):1207–1210, 2006.

[XBL05] L. Xiao, S. Boyd, and S. Lall. A scheme for asynchronous distributed sensor fusion based on average consensus. In *Symposium on Information Processing of Sensor Networks (IPSN)*, pages 63–70, Los Angeles, CA, April 2005.

[ZP05] M. M. Zavlanos and G. J. Pappas. Controlling connectivity of dynamic graphs. In *IEEE Conf. on Decision and Control and European Control Conference*, pages 6388–6393, Seville, Spain, December 2005.

[ZP07] M. Zavlanos and G. Pappas. Dynamic assignment in distributed motion planning with local information. In *American Control Conference*, pages 1173–1178, New York, July 2007.