Communication Complexity of Multi-Robot Systems

Eric Klavins *
California Institute of Technology, Pasadena, CA
klavins@caltech.edu

We examine the scalability of multi-robot algorithms. In particular, we wish to capture the idea that the less coordination a multi-robot system requires, the better it should scale to large numbers of robots. To that end, we introduce a notion of communication complexity *of multi-robot (or more generally, distributed control) systems as a surrogate for coordination. We describe a formalism, called DRL, for specifying multi-robot systems and algorithms for which the definition of communication complexity arises naturally. We then analyze the communication complexity of several, in some cases novel, multi-robot communication schemes each representative of one of several natural complexity classes.*

1 Introduction

Research in multi-robot systems has matured to the point where systems with tens, hundreds or even thousands of robots are being proposed. Usually, to achieve a given task, the robots must share information — about what they are sensing, for example. Because more sharing requires more resources (time, sensory effort and communication bandwidth), the amount of information that must be shared determines how *coordinated* a task is.

The adverse effects of coordination were perhaps first noticed by researchers designing algorithms for parallel processing computers: Often, increasing the number of processors results in a corresponding decrease in the amount of time needed to perform a given computation only up to a point, after which the time spent coordinating inter-process communication outweighs the benefits of having more processors.

In contrast, multiple robots are not usually used to increase the speed of a computation. Rather, such systems are designed to cover a large area with sensors; or achieve fault tolerance by using large numbers of cheap, expendable units; or manipulate an object in parallel with vast numbers of tiny, relatively independent actuators. Nevertheless roboticists are faced with problems similar to those of the parallel algorithm designer: If the task charged to a multi-robot system by necessity requires a great deal of coordination, then the performance of the system will necessarily degrade as more robots are employed.

Thus a measure of *coordination complexity* is needed to evaluate proposed algorithms. Ideally such a measure should be a function of the minimal *information flow* required to perform a given task and be independent of how that flow is mediated (by sensors, communication networks, etc.). It should, furthermore, account for how that information changes (becomes obsolete) as a result of the dynamics of the environment. We do not yet have a way to characterize arbitrary information flow, however, so we use the more basic notion of communication complexity (Definition 2.2) to capture coordination complexity. The model we assume is that each robot knows its own state and can communicate directly with another other robot via some (unspecified) communication system. The cost of a communication event, in terms of bandwidth used, latency or time spent waiting, is summarized into a single, abstract cost (denoted by γ in Definition 2.2). At every step of a multi-robot algorithm, we sum the costs of the communication events that occurred in that step and then take the average over all the (infinitely many) steps in the control algorithm.

With this definition, it turns out that the worst case complexity is $O(n^2)$: all robots must communicate constantly with all other robots. With $O(n^2)$ complexity, the bandwidth of the communication system must increase with the square of the number of robots. For this reason, an $O(n^2)$ complexity algorithm or task is not considered scalable. In an $O(n)$ complexity algorithm, on the other hand, bandwidth need only scale linearly with the number of robots. In this case we imagine that, for example, for every 100 robots, we add one

*This research is supported in part by the DARPA SEC program under grant number F33615-98-C-3613 and by AFOSR grant number F49620-01-1-0361.

new communication node, and do not suffer a decrease in performance.

Ultimately, the question we would like to answer is: Given a task what is the minimum amount of communication required to achieve the task — for any number of robots? Such results are usually difficult to obtain in complexity theory. Thus, for now we examine algorithms we believe to be representative of various natural complexity classes, leaving the above question to future work. In particular, we explore communication schemes that depend on some minimal information about the motion of the robots – for example, how the distances between robots change. These schemes are intended to be used as components of higher level control algorithms, which we do not address here.

The communication schemes we investigate range from full communication to no communication. Two of the schemes are new. The first, *Distance Modulated Communication (DMC)*, uses the idea that a robot might need a very accurate estimate of the position of nearby robots (to avoid colliding with them, for example) while needing only a coarse estimate of more distant robots. Thus, the scheme has any two robots communicating at a frequency proportional to the distance between them. It is shown that DMC has communication complexity $O(n \log n)$ or $O(n^{1.5})$ depending on assumptions about the dynamics of the robots. The second, the *Wandering Communication Scheme (WCS)*, defines a protocol by which only a (small) constant number of "wandering" robots are allowed to move and communicate, while others must remain immobile. A wanderer may transfer its right to move, and its information about the world, to an immobile robot in a short burst of communication. It is shown that WCS has communication complexity $O(n)$ or $O(1)$ depending on assumptions about how certain higher level decisions are made by the robots.

In this paper, systems are specified in a variant of the UNITY language [1] which we call DRL (for Distributed Robot Language) [6]. UNITY is an increasingly popular language for describing parallel algorithms in a way amenable to analysis. DRL is quite similar to UNITY, except that it is slightly more general and it is modified for real time systems. In DRL, it is easy to both model the environment (e.g. the physics of the robotic systems involved) and represent the algorithms that operate on the environment. The definition of communication complexity (Definition 2.2), which is similar to that used in parallel algorithms, arises quite naturally in DRL. Had we not been investigating DRL for its use in verifying multi-robot systems [6], we may not have arrived at Definition 2.2. However, the definition can certainly be adapted to other formalisms. The reader is advised, in this paper, to consider DRL as merely a convenient and formal specification language.

Specific Contributions The specific contributions of this paper are as follows. In Section 2, the specification language DRL [6] is augmented with a notion of communication cost and the formal notion of communication complexity is defined. We also supply a natural way to represent the dynamics of a multi-robot system. We then describe, in Lemmas 5, 6 and 7, several convenient tools for determining the communication complexity of a specification from an understanding of the frequency with which certain rules in the specification are applied. These lemmas are used in Section 3 to determine the communication complexity of various multi-robot communication schemes that we have borrowed or devised so that a member of each natural complexity class ($O(n^2)$, $O(n^{1.5})$, $O(n \log n)$, $O(n)$ and $O(1)$) is represented. Several of the schemes in Section 3 are quite straightforward, while two — the Distance Modulated Communication scheme and the Wandering Communication Scheme — are, to the best of our knowledge, new and presented in this paper for the first time.

Related Work We are primarily interested in scalable algorithms for formation forming [3] and other cooperative tasks for multi-vehicle systems [2] as well as distributed and self assembly [5]. The specification language DRL that we describe in Section 2.1 is very similar to the UNITY language [1]. Our main goal with DRL (not addressed here) is to develop a modeling, synthesis and verification tool for multi-robot and decentralized control systems, as described in [6]. Our notion of communication complexity borrows heavily from notions of communication complexity defined in the analysis of parallel algorithms as in, for example, [8] — the difference being that in the control systems we design we are concerned with maintaining some invariant indefinitely, as opposed to performing some finite computation. The idea of examining communication complexity was inspired by, but only superficially related to, the sort of communication complexity developed in [7]. Bandwidth-aware control and communication schemes are receiving increased attention in the controls literature [9], but scalabilty as addressed in this paper has not been addressed elsewhere.

2 Definitions

2.1 DRL: A Specification Language

We describe DRL briefly here. We suppose we have a set V of **variable symbols** and define a **state** s to be a function from V into some universe U of values: $s : V \to U$. We let S denote the set of all states. A *specification* is a means of describing what sequences of states we would like to allow. We do this with state transformers called *clauses*.

Definition 1 A **clause** is a pair $g : r$ where $g : S \to \texttt{Bool}$ is called the **guard** and $r : S \times S \to \texttt{Bool}$ is called the **rule**. If c is a clause, then the guard of c is denoted $c.g$ and the rule is denoted $c.r$.

Guards are expressed syntactically as logical expressions over V and rules are expressed syntactically as logical expressions over $V \cup V'$ where $V' = \{v' : v \in V\}$ is the set of *primed* variable symbols from V. A primed variable symbol refers to the next value of a variable while an unprimed variable symbol refers to the present value. For example, if c is the clause

$$x > 0 : y' = z \wedge x' < 0$$

then $c(s_1, s_2)$ essentially means

$$(s_1(x) > 0 \to s_2(y) = s_1(z) \wedge s_2(x) > 0)$$

$$\wedge\ (s_1(x) \leq 0 \to s_2 = s_1).$$

A variable $v \in V$ is said to **occur** in a clause $g : r$ if v appears in the expression for either g or r. It is said to **occur primed** in $g : r$ if v' occurs in r. If $c.g(s) = \mathit{true}$ we say that c is **applicable** in state s.

A specification is essentially a collection of clauses, along with an initial condition, a special clause that describes the dynamics of the environment, and a communication cost function.

Definition 2 A **specification** is a quadruple $\Pi = (\mathcal{I}, \mathcal{C}, \Delta, \gamma)$ where

1. $\mathcal{I} : S \to \texttt{Bool}$ is the **initial condition**
2. $\mathcal{C}$ is a set of clauses
3. Δ is a clause called the **dynamics** clause
4. $\gamma : \mathcal{C} \to \mathbb{R}^+ \cup \{0\}$ gives the **communication cost** of each clause.

This leads to the notion of an execution of a specification.

Definition 3 Let $\Pi = (\mathcal{I}, \mathcal{C}, \Delta, \gamma)$ be a specification. An execution of Π is a sequence $\{s_k\}$ for $k \in \mathbb{N}$ such that

1. $\mathcal{I}(s_0)$
2. For all $k \geq 0$
   (a) for all $c \in \mathcal{C}$, if $c.g(s_k)$ then $c.r(s_k, s_{k+1})$
   (b) $\Delta.r(s_k, s_{k+1})$
   (c) if $v \in V$ does not occur primed in any clause applicable in state s_k, then $s_{k+1}(v) = s_k(v)$.

We denote the set of all executions of Π by $\mathcal{E}(\Pi)$.

2.2 Communication Complexity of Specifications

We have already defined the cost of a clause. We next define the *communication complexity* of a step in an execution, of an execution, and finally of a specification. For each of these notions of complexity we use the polymorphic symbol $cc(\cdot)$, the exact meaning of which will always be clear from context.

Definition 4 Assume a fixed specification $\Pi = (\mathcal{I}, \mathcal{C}, \Delta, \gamma)$ is given. The **communication complexity** of a *state* $s \in S$ is

$$cc(s) \triangleq \sum_{c \in \mathcal{C} \wedge c.g(s)} \gamma(c).$$

The **communication complexity** of an *execution* $\{s_k\} \in \mathcal{E}(\Pi)$ is

$$cc(\{s_k\}) = \lim_{T \to \infty} \frac{1}{T} \sum_{k=1}^{T} cc(s_k).$$

The **communication complexity** of the *specification* Π is

$$cc(\Pi) = \max_{\{s_k\} \in \mathcal{E}(\Pi)} cc(\{s_k\}).$$

Thus, $cc(s)$ is sum of the costs of all clauses applicable s; $cc(\{s_k\})$ is the infinite average of the complexity of each step in the execution $\{s_k\}$; and $cc(\Pi)$ is equal to the worst case communication complexity of any execution of Π.

2.3 Basic Properties and Proof Methods

We next supply some basic results that often aid in determining the communication complexity of a particular specification. Generally, to prove that a given specification has a certain communication complexity, we determine the frequency with which each clause is applicable in any execution. This may be done deterministically (Lemma 5) or probabilistically (Lemmas 6 and 7). The proofs of these properties are given in the Appendix to this paper. The first lemma allows us to easily compute the cost of a specification that has periodic executions.

Lemma 5 *Let $\Pi = (\mathcal{I}, \mathcal{C}, \Delta, \gamma)$ be a specification and suppose that each clause $c_i \in \mathcal{C}$ is applicable every r_i steps in any execution of Π. Then*

$$cc(\Pi) = \sum_{i=1}^{|\mathcal{C}|} \frac{\gamma(c_i)}{r_i}.$$

Next suppose that the cost of each step k of any execution is modeled by a non-negative real valued random variable C_k with expectation $E[C_k]$. In the proofs of Theorems 10 and 14, C_k is essentially a function of a probabilistic model of the frequencies with which clauses in the corresponding specifications are applicable. The probabilistic models arise from either assumptions about the relative locations of the robots (e.g. their locations may be uniformly distributed in their workspace) or assumptions on the frequency which which certain guards are true (e.g. a guard may be true at some step if a flip of a biased coin comes up heads at that step).

Given C_k, we would like to compute the expected cost of the specification which, in general, is

$$E[cc(\Pi)] = \lim_{T\to\infty} \frac{1}{T} \sum_{k=1}^{T} E[C_k].$$

We begin with the case where each C_k is independent and identically distributed.

Lemma 6 *Let $\Pi = (\mathcal{I}, \mathcal{C}, \Delta, \gamma)$ be a specification and suppose the cost of executing each step k is modeled by the random variable C. Then*

$$E[cc(\Pi)] = E[C].$$

The next lemma allows us to compute the communication complexity of a specification if we know what the limit of the step cost is. It will be useful when we consider specifications modeled by Markov chains whose k-step transition probabilities are known (as in Theorem 14). Of course, Lemma 6 is a special case of Lemma 7.

Lemma 7 *Let $\Pi = (\mathcal{I}, \mathcal{C}, \Delta, \gamma)$ be a specification and suppose the cost of each step k in any execution is C_k. If*

$$\lim_{k\to\infty} C_k = \kappa$$

then $cc(\Pi) = \kappa$. Furthermore, if each C_k is a random variable and

$$\lim_{k\to\infty} E[C_k] = \kappa,$$

then $E[cc(\Pi)] = \kappa$.

Remark: Observe that the definition of communication complexity is an infinite average. A finite, initial period of $O(n^2)$ communication will not affect an ultimately less complex algorithm. Thus, in the specifications in this paper, we may assume that the global state is known initially since this could have been obtained by a brief period of total communication.

2.4 Specifications of Multi-Robot Systems

In the sequel we consider systems with n robots with locations $(x_i, y_i) \in \mathbb{R}^2$ for each $i \in \{1, ..., n\}$. We suppose (implicitly) that certain variables in V and certain clauses in $\mathcal{C}$ are *owned* by particular robots. Variables owned by robot i are denoted as such via an initial subscript i, as in x_i or $e_{i,j}$. The cost $\gamma(c)$ of a clause c is generally the result of an explicit reference to a variable v_j in a clause owned by robot i where $i \neq j$. That is, a reference to a variable owned by another robot incurs some cost to communicate the value of the variable. Variables not sub-scripted are not owned (i.e. they are global) and the dynamics clause has no cost, being executed, as it is, by the environment (which by definition includes all of the robots).

In the dynamics clause we will generally be noncommittal about the details of the system specified. For example, we will make use of the clause $\Delta(n)$ defined by

$$true : t' = t + \delta \wedge \forall i. \, (||x_i' - x_i|| < \delta v_{max}) \qquad (1)$$

that simply states time goes forward by some constant $0 < \delta \ll 1$ and the velocity of any robot is bounded by some constant $v_{max} > 0$.[1] Note that we have defined a family of dynamics clauses, one for each $n \in \mathbb{N}$ and

have used the parameter n implicitly in the rule for $\Delta(n)$ by assuming that i is quantified over $\{1, ..., n\}$.

3 Multiple Robot Communication Schemes

In this section we present multi-robot communication schemes for several natural complexity classes: $\mathrm{O}(n^2)$, $\mathrm{O}(n^{1.5})$, $\mathrm{O}(n \log n)$, $\mathrm{O}(n)$ and $\mathrm{O}(1)$. We usually assume the dynamics of the robots are given by (1) or some variant of it. A discussion of how these schemes may be used, augmented and extended appears in Section 4. The scheme we omit is the empty communication scheme where no communication occurs at all. This scheme clearly has communication complexity $O(0)$.

3.1 The Worst Case: $\mathrm{O}(n^2)$ Complexity

Many simple multi-robot algorithms really treat a group of robots as a single (albeit disconnected) robot. Assuming there is no leader (a fragility one might wish to avoid), such algorithms usually require that a good estimate of the global state of the system be known by each robot. Thus, each robot i must maintain, for each j, an estimate of x_j which we denote $e_{i,j}$. Furthermore we require that the property

$$||e_{i,j} - x_j|| < \varepsilon \tag{2}$$

hold at any time, for some constant $\varepsilon > \delta v_{max}$ which may be, for example, required for the stability of whatever control algorithm the system is using. If we assume the dynamics are given by (1), the best that can be achieved is for each robot i to include the clause $c_{i,j}$ defined by

$$(t + \delta - l_{i,j})v_{max} > \varepsilon \ : \ e'_{i,j} = x_j \wedge l'_{i,j} = t \tag{3}$$

for $i, j \in \{1, ..., n\}$ and $i \neq j$. We set $\gamma(c_{i,j}) = 1$. In (3), $l_{i,j}$ represents the time at which the last update to $e_{i,j}$ took place. The inequality in the guard assumes the worst case movement (a velocity of exactly v_{max}) by robot j — that is, $(t + \delta - l_{i,j})v_{max}$ represents the worst case value for $||e_{i,j} - x_j||$ in the next step (at time $t + \delta$). If this exceeds ε, then the estimate $e_{i,j}$ and the time of last update $l_{i,j}$ are updated.

Supposing that the guard in $c_{i,j}$ is an equality and solving for $t - l_{i,j}$ gives that the clause will be applied every $r_{i,j} = \frac{\varepsilon - \delta v_{max}}{v_{max}}$ steps. Using Lemma 5 we obtain the simple result

Theorem 8 *Let*

$$\mathcal{I}(n) = \forall i, j \ . \ e_{i,j} = x_j \wedge l_{i,j} = 0$$

and

$$\mathcal{C} = \{c_{i,j} \mid 1 \leq i \neq j \leq n\}$$

and put

$$\Pi_{global}(n) = (\mathcal{I}(n), \mathcal{C}(n), \Delta(n), \gamma)$$

where $c_{i,j}$ is as in (3) and Δ is as in (1). Then $cc(\Pi_{global}(n)) = \mathrm{O}(n^2)$.

We use the fact that $|\mathcal{C}| = n(n-1)$. Note that the result is independent of the constants δ, v_{max} and ε which essentially determine (the constant) $r_{i,j}$. Thus, even techniques which increase the period $r_{i,j}$ of communication significantly, by having each robot use a sophisticated estimator of all other robot's positions, for example [9], ultimately do not scale in the sense that we have introduced in this paper.

3.2 Sub-$\mathrm{O}(n^2)$ Complexity

In this section we investigate a specification for *distance modulated communication* (DMC). The specification, shown in Figure 1, is based on the idea that closer robots should communicate their positions more frequently than distant ones. The goal is to maintain the invariant

$$||e_{i,j} - x_j|| \leq k||x_i - x_j||$$

for each $i \neq j$. Here $e_{i,j}$ is the ith robot's estimate of the jth robot's position and $k > 0$ is a constant. In Figure 1, the behavior of each robot i is specified by the clause $c_{i,j}$, for $j \in \{1, ..., n\} - \{i\}$. The variable $l_{i,j}$ is used by robot i to keep track of the last time robot i updated its estimate $e_{i,j}$, thereby accruing 1 unit of communication cost. The quantity $(t - l_{i,j})v_{max}$ in the guard represents the maximum possible difference between $e_{i,j}$ and x_j. The quantity $||e_{i,j} - x_i|| - (t - l_{i,j})v_{max}$ represents the minimum possible distance between the two robots. That is

$$||e_{i,j} - x_j|| \leq (t - l_{i,j})v_{max}$$

and

$$||e_{i,j} - x_i|| - (t - l_{i,j})v_{max} \leq ||x_i - x_j||$$

[1] This constraint on the velocity can be considered as a crude discretization of the actual continuous dynamics. We assume that $1/\delta$ is greater than the frequency of the actual dynamics so that aliasing problems do not occur.

Define $c_{i,j}$ by

$$(t - l_{i,j})v_{max} \geq k\,(||e_{i,j} - x_i|| - (t - l_{i,j})v_{max}) : e'_{i,j} = x_j \wedge l'_{i,j} = t$$

$$\gamma(c_{i,j}) = 1$$

$$\mathcal{I}(n) \equiv e_{i,j} = x_j \wedge l_{i,j} = 0 \wedge t = 0$$

$$\mathcal{C}(n) = \{c_{i,j} \mid i, j \in \{1, ..., n\} \wedge i \neq j\}$$

Then $\Pi_{DMC}(n) = (\mathcal{I}(n), \mathcal{C}(n), \Delta(n), \gamma)$

Fig. 1: The specification of the distance modulated communication scheme Π_{DMC}. $\Delta(n)$ refers to a dynamics clause such as that in equation (1).

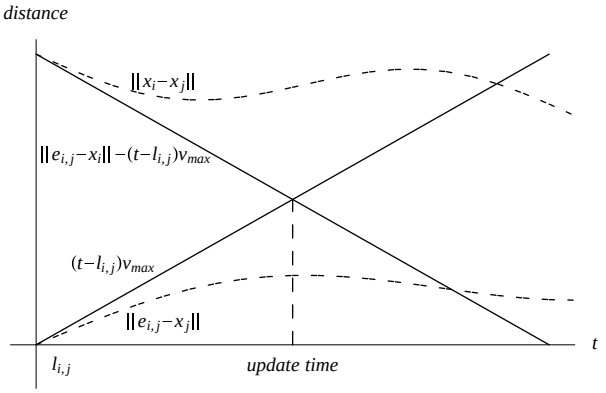

Fig. 2: Estimates used in the DMC specification.

as illustrated in Figure 2. Each of these bounds can be computed by agent i without reference to x_j and so require no communication cost. When the bound on the estimate error exceeds k times the bound on the distance, the guard becomes true and $e_{i,j}$ and $l_{i,j}$ are updated.

The communication complexity of Π_{DMC}, of course, depends on the relative locations of the robots while the specification is enforced. We will show that under two different reasonable assumptions about the robot locations, that $cc(\Pi_{DMC}(n))$ is better than $\mathrm{O}(n^2)$. First we suppose that the robots are arranged in a line. We specify this in the initial condition and by using a trivial dynamics clause. Of course, the theorem holds for any spacing and any dynamics clause that (even approximately) preserves it, and the extremely simple dynamics clause in the statement of the theorem below is used merely for simplicity in presentation.

Theorem 9 *Let*

$$\mathcal{J}(n) = \forall i \in \{1, ..., n\}(x_i = (i, 0))$$

and

$$\Delta = true : t' = t + \delta$$

and put

$$\Pi_{DMC}(n) = (\mathcal{I}(n) \wedge \mathcal{J}(n), \mathcal{C}(n), \Delta, \gamma)$$

where $\mathcal{I}(n)$, $\mathcal{C}(n)$ and γ are as in Figure 1. Then $cc(\Pi_{DMC}(n)) = \mathrm{O}(n \log n)$.

Proof: Let $d_{i,j} \triangleq ||x_i - x_j|| = |i - j|$. Solving

$$tv_{max} = k(d_{i,j} - tv_{max})$$

for t shows that clause $c_{i,j}$ will be applicable every

$$r_{i,j} \triangleq \frac{kd_{i,j}}{(1 + k)\delta v_{max}} = \alpha|i - j|$$

steps where $\alpha \triangleq \frac{k}{(1+k)\delta v_{max}}$. By Lemma 5, the communication complexity is thus

$$cc(\Pi_{DMC}(n)) = \sum_{1 \leq i \neq j \leq n} \frac{1}{\alpha|i - j|} = \frac{2}{\alpha} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{1}{j - i}.$$

It is straightforward to show that this sum evaluates to

$$\frac{2}{\alpha}\,(nH(n) - n) = \mathrm{O}(n \log n)$$

where $H(n)$ is the nth harmonic number. □

We next explore the somewhat more natural assumption that the robots are randomly distributed in a square on the plane with an average density of ρ robots per square meter and determine the *expected value* of the communication complexity of $\Pi_{DMC}(n)$. We therefore suppose that the position of robot i is given by a pair of independent identically distributed random variables, (X_i, Y_i), with the common density function

$$f_m(x) \triangleq \begin{cases} \frac{1}{2m} \text{ if } x \in [-m, m] \\ \\ 0 \text{ otherwise.} \end{cases}$$

For some $m \in \mathbb{R}$ yet to be determined. We then have

Theorem 10 *Let*

$$\mathcal{J}(n) = \forall i \in \{1, ..., n\}(x_i \in [-m, m])$$

where $m = \sqrt{\frac{n}{4\rho}}$ *and let* Δ *be as in Theorem 9. Let*

$$\Pi_{DMC}(n) = (\mathcal{I}(n) \wedge \mathcal{J}(n), \mathcal{C}(n), \Delta(n), \gamma)$$

where $\mathcal{I}(n)$, $\mathcal{C}(n)$ *and* γ *are as in Figure 1. Then* $E[cc(\Pi_{DMC}(n))] = \mathrm{O}(n^{1.5})$.

This result suggests that a uniform distribution of robots in a square of density ρ is less sparse than an equal spacing of the robots in a line. In fact, one expects that the complexity of Π_{DMC} approaches $O(n^2)$ as the dimension of the workspace grows.

Proof: For each $i \neq j$, define the random variable $D_{i,j} = ||X_i - X_j||$. Then clause $c_{i,j}$ will be applicable every $R_{i,j} = \alpha D_{i,j}$ steps, where α is the same as in the proof of Theorem 9. The cost of a single step is then

$$C \triangleq \sum_{1 \leq i \neq j \leq n} \frac{1}{R_{i,j}}$$

units. Using Lemma 6 we have that

$$E[cc(\Pi_{DMC}(n))] = E[C].$$

Next we make use of the following lemma (proved in the Appendix).

Lemma 11 *The expected cost at each step is*

$$E[C] = \frac{\beta}{\alpha} \frac{1}{16m} n(n-1)$$

where $\beta = \frac{8}{3}(4 - 4\sqrt{2} - 9\ln(\sqrt{2} - 1) + 3\ln(\sqrt{2} + 1))$.

The area of the square in which the robots are placed is $4m^2$. Taking $m = \sqrt{\frac{n}{4\rho}}$ gives an area of n/ρ so that there are ρ robots per square meter. We then have

$$E[cc(\Pi_{DMC}(n))] = E[C] = 2\frac{\beta\sqrt{\rho}}{16\alpha}\sqrt{n}(n-1) = \mathrm{O}(n^{1.5})$$

as desired. □

The conditions under which the DMC protocol can be used are much broader than the two situations we have analyzed in Theorems 9 and 10. However, we believe these two cases to be typical. The first represents a common *formation* and other formations likely produce similar complexity results. The second represents the reasonable assumption that many multi-robot algorithms would tend to keep the robots more or less evenly distributed in the workspace: i.e. that the workspace would grow if the number of robots were increased.

3.3 $\mathrm{O}(n)$ Complexity

In [8] it is shown (in a slightly somewhat different model than that presently considered) that the communication complexity of systems with n processors arranged in a graph with constant degree is $\mathrm{O}(n)$. We can easily obtain a similar result here with respect to multi-robot systems. In particular, suppose we have a graph $G = (\{1, ..., n\}, E)$ where $E \subseteq V \times V$ and that we wish to maintain the property

$$(i, j) \in E \Rightarrow ||e_{i,j} - x_j|| < \varepsilon$$

in place of (2). Then the obvious corollary can be shown:

Corollary 12 *Let*

$$\mathcal{I}(n) = \forall i, j \; . \; e_{i,j} = x_j \wedge l_{i,j} = 0$$

and

$$\mathcal{C} = \{c_{i,j} \; | (i, j) \in E\}$$

and put

$$\Pi_{nbr}(n) = (\mathcal{I}(n), \mathcal{C}(n), \Delta(n), \gamma)$$

where $\Delta(n)$ *is as in (1) and* $c_{i,j}$ *is as in (3). Then* $cc(\Pi_{nbr}(n)) = \mathrm{O}(n)$.

Another commonly used scheme for controlling a group of robots is to have each robot transmit its location to a leader robot (or control computer) with some frequency. The leader then sends to each robot a command based on the collected global state. The communication complexity of this scheme is also $O(n)$, assuming the size of the commands does not depend on n. This scheme (in its simplest form) is not robust to failures in the leader, however.

3.4 Better Than $\mathrm{O}(n)$ Complexity

The obvious way to get better than $\mathrm{O}(n)$ complexity is to have no communication at all. The subject of how to do anything useful in this realm of operation (say by simple local sensing) has been investigated by others and is beyond the scope of this paper. Instead, in this section, we examine a task similar to that explored in Section 3.1: to maintain the invariant

$$||e_{i,j} - x_j|| < \varepsilon$$

for some constant ε for any *currently moving* robot. This is possible with lower than $\mathrm{O}(n^2)$ communication

For $i \in \{1, ..., n\}$ define clauses $c_{i,1}$, $c_{i,2}$ and $c_{i,3}$ by

$$
\begin{array}{l}
q_i = 1 : \forall j (q_j \neq 3 \rightarrow e'_{i,j} = x_j) \\
q_i = 1 \wedge \mathit{coin}(p,t) : q'_i = 2 \\
q_i = 2 : q'_i = 3 \wedge r'_i = \perp \wedge \\
\qquad \exists j \left[q_j = 3 \wedge q'_j = 1 \wedge r'_j = i \wedge \forall k (e'_{j,k} = e_{i,k}) \right]
\end{array}
$$

respectively. Then set

$$
\begin{array}{l}
\gamma(c_{i,1}) = \kappa - 1 \\
\gamma(c_{i,2}) = 0 \\
\gamma(c_{i,3}) = n
\end{array}
$$

$$
\begin{array}{l}
\mathcal{I}(n) \equiv \forall i [(i \leq \kappa \rightarrow q_i = 1) \wedge (i > \kappa \rightarrow q_i = 3) \\
\qquad \wedge r_i = \perp \wedge \forall j (e_{i,j} = x_j)]
\end{array}
$$

$$
\mathcal{C}(n) = \bigcup_{i=1}^{n} \{c_{i,1}, c_{i,2}, c_{i,3}\}
$$

$$
\begin{array}{l}
\Delta(n) = \mathit{true} : t' = t + \delta \wedge \\
\quad \forall i [(q_i = 1 \rightarrow ||x'_i - x_i|| \leq \delta v_{max} \\
\quad \wedge (q_i \neq 1 \rightarrow x'_i = x_i)]
\end{array}
$$

$$
\Pi_{WCS}(n) = (\mathcal{I}(n), \mathcal{C}(n), \Delta(n), \gamma)
$$

Fig. 3: The specification of the Wandering Communication Scheme (WCS). $\mathit{coin}(p,t)$ is a random number generator that is true at time t with probability p. We assume qualification over $\{1, ..., n\}$ in the notation.

complexity if we restrict the motions of the robots, allowing only certain robots to move at any given time (the dynamics clause in Figure 3, for example, enforces this constraint). In particular, we present a communication and movement scheme that has either O(n) or O(1) communication complexity for any number of robots, depending on our assumptions. The scheme takes advantage of the presumption that any robot that is not moving does not need to know any other robot's states. In particular, we choose a constant κ and only allow, for any number of robots n, that κ of them can move at any time. A robot can give up its right to move by stopping and uploading its estimates of the other robot locations to a newly chosen and formerly idle robot.

The specification of the scheme, called the *Wandering Communication Scheme* (WCS), is shown in Figure 3. The meanings of the variables are as follows. The variable q_i denotes the state of the robots as either wandering (1), uploading (2) or idle (3). The variable $e_{i,j}$ is the ith robot's estimate of x_j. Setting the "reservation" variable r_i to j specifies that robot j is uploading to robot i and its appearance in $c_{i,3}$ prevents two robots from uploading to a same new robot.

Clause $c_{i,1}$ is used to keep a wandering robot's estimate of the other moving robots accurate. Clause $c_{i,2}$ is used to decide whether to switch from wandering to idle. It uses the random number generator $\mathit{coin}(p,t)$, which is true at time t with probability p. Clause $c_{i,3}$ is used by robot i to choose a new robot j, to whom it will hand over its "right" to move: uploading its state estimates to j and switching j's state to wandering and its own state to idle.

For a given n and κ we model an execution of $\Pi_{WCS}(n)$ as a Markov Chain whose states X_m correspond to the possible number of robots in each specification state $q_i \in \{1, 2, 3\}$ at step m. Now, the number of robots that are in state 3 is always $n - \kappa$. Thus X_m is just the number of robots in state 1. The number of robots in state 2 is just κ minus the number of robots in state 1. The transition probabilities are, for each i and j,

$$
\begin{array}{l}
P[X_{m+1} = \kappa - j \mid X_m = \kappa - i] \\
= \begin{cases}
\left(\begin{array}{c} \kappa - i \\ j \end{array} \right) p^j (1-p)^{\kappa - i - j} \text{ if } 1 \leq j \leq \kappa - i \\
0 \text{ otherwise.}
\end{cases}
\end{array} \tag{4}
$$

We define the $\kappa + 1$ dimensional transition matrix P to be the matrix whose jth row and ith column is given by (4).

Next, we define the cost of each step X_m. The communication cost of a robot in state 1 is $\kappa - 1$ and the the communication cost of a robot in state 2 is n. In state 3, a robot does not communicate. Thus, the cost at step m is

$$
C_m \triangleq (\kappa - 1) X_m + n(\kappa - X_m).
$$

And the expected total cost at step m is

$$
E[C_m] = (1 \underbrace{0 \ldots 0}_{\kappa \text{ times}}) P^m r \tag{5}
$$

where r is the cost vector

$$
r \triangleq \begin{pmatrix}
(\kappa - 1)\kappa + 0n \\
(\kappa - 1)(\kappa - 1) + 1n \\
\vdots \\
(\kappa - 1)0 + n\kappa
\end{pmatrix}.
$$

We show, by straightforward analysis (as described in many texts [4] on the subject) in the Appendix that

Lemma 13 *Let x be the $\kappa+1$ dimensional vector whose ith element is*

$$\left(\begin{array}{c} \kappa \\ i \end{array} \right) p^{i-\kappa}$$

for $i \in \{0,...,\kappa\}$. Then as m goes to infinity, P^m goes to the matrix whose rows are all given by

$$\tilde{x}^T \triangleq \left(\frac{p}{p+1}\right)^{\kappa} x^T.$$

Using this result we can show that

Theorem 14 *If the value of p in $\Pi_{WCS}(n)$ is constant, then*

$$E[cc(\Pi_{WCS}(n))] = \mathrm{O}(n)$$

and if $p \triangleq 1/n$ then

$$E[cc(\Pi_{WCS}(n))] = \mathrm{O}(1)$$

Proof: The limit of the expected value of the cost (5) as $m \to \infty$ is

$$\tilde{x}^T r = \left(\frac{p}{1+p}\right)^{\kappa} \sum_{j=0}^{\kappa} [(\kappa-1)j + (\kappa-j)n] \left(\begin{array}{c} \kappa \\ j \end{array} \right) p^{j-k}$$

$$= \frac{\kappa}{1+p}(\kappa - 1 + np).$$

We now use Lemma 7 to conclude that if p is constant, that $E[cc(\Pi_{WCS}(n))] = \mathrm{O}(n)$. If, on the other hand, $p \triangleq 1/n$, then

$$\lim_{m\to\infty} E[C_m(n)] = \frac{\kappa}{1+\frac{1}{n}}\kappa < \kappa^2 = \mathrm{O}(1).$$

Once again using Lemma 7 yields the desired result. □

4 Discussion

Control: Pairing a communication scheme with a high level controller depends on the requirements of the controller and the guarantees of the the scheme. Using DMC, for example, a reactive planner for each robot may be constructed as follows. Consider two robots with indices i and j and suppose that i has last communicated with j at time $l_{i,j}$. It can easily be determined, from the guard in clause $c_{i,j}$ in Figure 1, at what time $l_{i,j} + \Delta t$ it will communicate with j again. Thus, vehicle i can simply avoid the region

$$R_j = \{\ x\ :\ ||e_{i,j} - x|| < \Delta t\, v_{max}\ \}$$

while planning its route through the workspace. Furthermore, to make the path planning problem easier, other distance metrics can be used in DMC such as the distance between robot j and the line segment connecting robot i with its goal position.

Assumptions: The analyses in Section 3 make assumptions that are certainly not as general as they could be. For example, Theorem 9 should hold for any formation that increases in size linearly with the number of robots used. Theorem 9 could be extended to allow motions that are guaranteed to preserve a certain expected distance between robots. The WCS has many variants, each of which affect the expected communication complexity. For example, constant communication complexity can be obtained by requiring that all idle robots be positioned in one of some large number of (sparsely placed) "island regions". The wandering robots wouldn't necessarily know where an idle robot was, but they would know at least that it was located in *some* island region. The transfers of position estimates in clause $c_{i,3}$ in Figure 3 would then have cost κ resulting in O(1) complexity overall.

Sensing: Although sensing is not really mentioned in Section 3, it is modeled. For example, the assumption that robots know their own locations implies a certain amount of sensing. Also, clause $c_{i,3}$ where it states "$\exists j[q_j = 3...$" assumes that robot i can find another robot whose state is 3 (idle). This information can not be obtained from the scheme itself and adding the communication required to make the discrete states of the robots global would most likely increase the communication complexity of the algorithm. The original intent, however, was to have wanderers *find* idle robots using their local sensors and then upload their estimates to them. More generally, however, incorporating sensing assumptions into DRL specifications remains an art and no clear path toward investigating, say, sensing vs. communication trade-offs has yet emerged.

Variants: There are many possible variants of Definition 2.2. For example, suppose that the cost of communication γ were scaled by the distance between the communicating robots, as is the case with low power sensor nets, or "smart dust". With this definition, communicating indirectly via hops in a network instead of directly may give a lower overall complexity. A *power*

aware communication scheme could be devised that determines the lowest energy route from one robot to another based on current estimates (possibly obtained by a variant of DMC or WCS) of the vehicle locations.

5 Conclusion

In this paper we introduce a notion of communication complexity as a means of investigating the scalability of multi-robot algorithms in terms of how much coordination they require. We present several communication schemes, two of which are new (to the best of our knowledge) that maintain estimates of the global state of varying accuracies in the face of simply-modeled possible changes in the environment. By analyzing their complexities, we show that these schemes cover several natural communication complexity classes from $\mathrm{O}(n^2)$ communication to $\mathrm{O}(1)$ communication. It is our hope that techniques similar to these will be used to check the scalability of newly proposed multi-robot or distributed control algorithms.

We have not included a model of possible sensing modalities and costs. Nor have we described the actual control algorithms and higher level planners that could be used in conjunction with the schemes we have presented. More importantly, we have only analyzed instances of algorithms that achieve a given task but have not answered the question: For a given task, what is the minimum communication complexity of *any* algorithm that achieves the task? Although this question appears to be quite difficult, we plan to attempt address it in a future paper, at least for very simple tasks.

A Appendix

A.1 Proof of Lemma 5

In T steps, clause i is executed $\frac{T}{r_i}$ times. Thus,

$$\sum_{k=1}^{T} cc(s_k) = T \sum_{i=1}^{|C|} \frac{\gamma(c_i)}{r_i}.$$

Multiplying by $1/T$ and taking the limit yields the desired result. □.

A.2 Proof of Lemma 6

Suppose the cost of each step in any execution of Π can be modeled as a random variable C and that the costs at each step are independent. Then

$$E[cc(\Pi)] = \lim_{T\to\infty} \frac{1}{T} \sum_{k=1}^{T} E[C] = \lim_{T\to\infty} \frac{1}{T} T E[C] = E[C]$$

as was to be shown. □

A.3 Proof of Lemma 7

Suppose

$$\lim_{k\to\infty} C_k = \kappa.$$

Then for any $\epsilon > 0$, a positive integer T_0 can be found so that $\kappa - \varepsilon < C_k < \kappa + \varepsilon$ for all $k > T_0$. Thus,

$$\frac{1}{T}\sum_{k=1}^{T} C_k = \frac{1}{T}\left(\sum_{k=1}^{T_0} C_k + \sum_{k=T_0+1}^{T} C_k\right)$$

$$< \frac{1}{T}\left[\alpha + (T - T_0 - 1)(\kappa + \varepsilon)\right]$$

where $\alpha = \sum_{k=1}^{T_0} C_k$. Similarly,

$$\frac{1}{T}\sum_{k=1}^{T} C_k > \left[\alpha + (T - T_0 - 1)(\kappa - \varepsilon)\right].$$

Taking limits, we arrive at

$$\kappa - \varepsilon < \lim_{T\to\infty} \frac{1}{T}\sum_{k=1}^{T} C_k < \kappa + \varepsilon.$$

This argument holds for any ε and thus we may conclude, by the *squeeze theorem*, that

$$\lim_{T\to\infty} \frac{1}{T}\sum_{k=1}^{T} C_k = \kappa.$$

The rest of the lemma follows easily. □

A.4 Proof of Lemma 11

Let

$$g(x_1, y_1, x_2, y_2) = \left((x_2 - x_1)^2 + (y_2 - y_1)^2\right)^{-\frac{1}{2}}.$$

Then straightforward integration shows that

$$\int_{-m}^{m}\int_{-m}^{m}\int_{-m}^{m}\int_{-m}^{m} g(x_1, y_1, x_2, y_2)\, dx_1 dy_1 dx_2 dy_2$$

$$= \frac{8}{3}m^3\left(4 - 4\sqrt{2} - 9\ln(\sqrt{2} - 1) + 3\ln(\sqrt{2} + 1)\right) \triangleq \beta m^3.$$

The expected value $E[C]$ is given by

$$\frac{1}{\alpha}\left(\frac{1}{2m}\right)^{2n} \underbrace{\int_{-m}^{m}\cdots\int_{-m}^{m}}_{n(n-1)\text{ times}} \sum_{1\le i\neq j\le n} g(x_i,y_i,x_j,y_j)\,dx_1...dy_n$$

$$= \frac{1}{\alpha}\left(\frac{1}{2m}\right)^{2n} \sum_{1\le i\neq j\le n} \underbrace{\int_{-m}^{m}\cdots\int_{-m}^{m}}_{n(n-1)\text{ times}} g(x_i,y_i,x_j,y_j)\,dx_1...dy_n$$

$$= \frac{1}{\alpha}\left(\frac{1}{2m}\right)^{2n} n(n-1) \underbrace{\int_{-m}^{m}\cdots\int_{-m}^{m}}_{n(n-1)-4\text{ times}} \beta m^3\,dx_1...dy_{n-2}$$

$$= \frac{1}{\alpha}\left(\frac{1}{2m}\right)^{2n} n(n-1)\beta m^3(2m)^{2n-4}$$

$$= \frac{1}{\alpha}\left(\frac{1}{2m}\right)^{4} n(n-1)\beta m^3 = \frac{\beta}{\alpha}\frac{1}{16m}n(n-1)\ \ \square$$

A.5 Proof of Lemma 13

It is known that the matrix P^T has 1 as an eigenvalue and that the steady state of the system $y^T P^m$ (for any given $y \neq 0$) is given by (any nonzero multiple of) the eigenvector corresponding to 1 [4]. Thus, we first show that

$$x^T P = x^T$$

is one such eigenvector,where x is as in the statement of the lemma. By the definition of matrix multiplication, the ith element of $x^T P$ is

$$\sum_{j=0}^{\kappa-i} \left(\begin{array}{c} k \\ j \end{array}\right) p^{j-\kappa} \left(\begin{array}{c} \kappa-j \\ i \end{array}\right) p^i (1-p)^{\kappa-i-j}.$$

Define $s \triangleq \kappa - i$. Then the sum becomes

$$p^{-s}\sum_{j=0}^{s} \left(\begin{array}{c} k \\ j \end{array}\right) \left(\begin{array}{c} \kappa-j \\ \kappa-s \end{array}\right) p^j (1-p)^{s-j}.$$

It is straightforward to show that sum part of this expression is $\left(\begin{array}{c} \kappa \\ s \end{array}\right)$. Thus, the ith element of $x^T P$ is

$$p^{-s}\left(\begin{array}{c} \kappa \\ s \end{array}\right) = \left(\begin{array}{c} \kappa \\ \kappa-i \end{array}\right) p^{i-\kappa} = \left(\begin{array}{c} \kappa \\ i \end{array}\right) p^{i-\kappa}$$

as desired.We now have that for any $y \neq 0$ that $y^T P^m \to \alpha x$ for some $\alpha \neq 0$. In particular,

$$(1\ 0\ ...\ 0)\lim_{m\to\infty} P^m = \alpha x$$

so that the first row of the limit of P^m is given by αx. Since each row of P^m must sum to 1, we obtain that

$$\alpha = \frac{1}{||x||} = \left(\sum_{i=0}^{\kappa}\left(\begin{array}{c} \kappa \\ i \end{array}\right) p^{i-\kappa}\right)^{-1} = \left(\frac{p}{p+1}\right)^{\kappa}$$

using the binomial theorem. The other rows of P^m are obtained similarly. $\square$.

References

[1] K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation*. Addision-Wesley, 1988.

[2] L. Cremean, B. Dunbar, D. van Gogh, J. Hickey, E. Klavins, J. Meltzer, and R. M. Murray. The Caltech multi-vehicle wireless testbed. In *Conference on Decision and Control*, 2002. Submitted for Review.

[3] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. In *IFAC World Congress*, 2002.

[4] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume I. Wiley, 1950.

[5] E. Klavins. Automatically synthesized controllers for distributed assembly: Partial correctness. In *Cooperative Control and Optimization*. Kluwer, Gainsville, FL, 2001.

[6] E. Klavins and J. Hickey. Specification and refinement of decentralized control systems. In *Conference on Decision and Control*, 2002. Submitted for Review.

[7] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[8] J. N. Tsitsiklis and G. D. Stamoulis. On the average communications complexity of asynchronous distributed algorithms. *Journal of the Association for Computing Machinery*, 42(2):382–400, March 1995.

[9] J. K. Yook, D. M. Tilbury, and N. R. Soparkar. Trading computation for bandwidth: Reducing communication in distributed control systems using state estimators. *IEEE Transactions on Control Systems Technology*, 10(5), July 2002.