A Geometric Assignment Problem for Robotic Networks

Stephen L. Smith and Francesco Bullo

Department of Mechanical Engineering
Center for Control, Dynamical Systems and Computation
University of California, Santa Barbara, CA 93106-5070, USA
stephen@engineering.ucsb.edu, bullo@engineering.ucsb.edu

Summary. In this chapter we look at a geometric target assignment problem consisting of an equal number of mobile robotic agents and distinct target locations. Each agent has a fixed communication range, a maximum speed, and knowledge of every target's position. The problem is to devise a distributed algorithm that allows the agents to divide the target locations among themselves and, simultaneously, leads each agent to its unique target. We summarize two algorithms for this problem; one designed for "sparse" environments, in which communication between robots is sparse, and one for "dense" environments, where communication is more prevalent. We characterize the asymptotic performance of these algorithms as the number of agents increases and the environment grows to accommodate them.

1 Introduction

Consider a group of n mobile robotic agents, equipped with wireless transceivers for limited range communication, dispersed in an environment $\mathcal{E} \subset \mathbb{R}^2$. Suppose the environment also contains n target locations, and each agent is given a list containing their positions (these positions may be given as GPS coordinates). We would like each target location to be occupied be an agent as quickly as possible. Since no *a priori* assignment of target-agent pairs has been given, the agents must solve the problem through communication and motion. We call this the *target assignment problem*. Such a problem could arise in several applications, such as UAV's on a surveillance mission, where the targets are the centers of their desired loitering patterns.

The centralized problem of simply assigning one agent to each target is known in the combinatorial optimization literature as the *maximum matching problem* [1]. There are several polynomial time algorithms for solving this problem, the best known being [2] by Hopcroft and Karp. To efficiently assign agents to targets, we may be interested in finding a maximum matching (i.e., an assignment of one agent to each target) which minimizes a cost function. If the cost function is the sum of distances from each agent to its assigned target, then the problem is known as the

assignment problem, or the *minimum weight maximum matching problem*, [1]. This problem can be written as an integer linear program and optimal solutions can be computed in polynomial time [3]. Another choice of cost function is to minimize the maximum distance between agents and their assigned targets. This problem is commonly referred to as the *bottleneck assignment problem* [4], and although the cost function is not linear, there still exist several polynomial time algorithms for its solution. There has also been work on developing algorithms for the assignment problem which can be implemented on parallel computing systems. One example is the *auction algorithm* [5], which can be implemented with one processor for each agent.

There is set of problems, commonly referred to as *decentralized task allocation*, that are closely related to our target assignment problem, see for example [6–8]. In these problems the goal is generally to assign vehicles to spatially distributed tasks while maximizing the "score" of the mission. Most works on this problem develop advanced heuristic methods, and demonstrate their effectiveness through simulation or real world implementation. In [9] the auction algorithm was adapted to solve a task allocation problem in the presence of communication delays. There has also been prior work on the target assignment problem [10–14]. For example, an algorithm based on hybrid systems tools is developed in [10]. The algorithm performance is characterized by a bound on the number of switches of the hybrid system; however, no analysis of the time complexity is provided.

In this chapter we summarize our recent investigations [12,13] into the minimum-time task assignment problem and its scalability properties. We are interested in characterizing the completion time as the number of agents, n, grows, and the environment, $\mathcal{E}(n) := [0, \ell(n)]^2$, grows to accommodate them. In Section 4 we describe the ETSP ASSGMT algorithm with worst-case completion time in $O(\sqrt{n}\ell(n))$. In addition, in "sparse" environments, i.e., when $\ell(n)/\sqrt{n}\theta_0 + \infty$, the ETSP ASSGMT algorithm is asymptotically optimal among a broad class of algorithms in terms of its worst-case completion time. Then, in Section 5 we describe the GRID ASSGMT algorithm with worst-case completion time in $O(\ell(n)^2)$. We also characterize the stochastic properties of the GRID ASSGMT algorithm in "dense" environments, i.e., when $\ell(n)/\sqrt{n}\theta_0 0$. If the agents and targets are uniformly randomly distributed, then the completion time belongs to $O(\ell(n))$ with high probability. Also, if there are n agents and only $n/\log n$ targets, then the completion time belongs to $O(1)$ with high probability.

The two algorithms are complementary: ETSP ASSGMT has better performance in sparse environments, while GRID ASSGMT has better performance in dense environments.

2 Geometric and stochastic preliminaries

In this section we review a few useful results on the Euclidean traveling salesperson problem, occupancy problems, and random geometric graphs. To do this, we must

first briefly review some notation. We let $\mathbb{R}$ denote the set of real numbers, $\mathbb{R}_{>0}$ denote the set of positive real numbers, and $\mathbb{N}$ denote the set of positive integers. Given a finite set A, we let $|A|$ denote its cardinality. For two functions $f, g : \mathbb{N}\theta_0\mathbb{R}_{>0}$, we write $f(n) \in O(g)$ (respectively, $f(n) \in \Omega(g)$) if there exist $N \in \mathbb{N}$ and $c \in \mathbb{R}_{>0}$ such that $f(n) \leq cg(n)$ for all $n \geq N$ (respectively, $f(n) \geq cg(n)$ for all $n \geq N$). If $f(n) \in O(g)$ and $f(n) \in \Omega(g)$ we say $f(n) \in \Theta(g)$. We say that event $A(n)$ occurs *with high probability* (w.h.p.) if the probability of $A(n)$ occurring tends to one as $n\theta_0 + \infty$.

2.1 The Euclidean traveling salesperson problem

For a set of n points, $\mathcal{Q} \in \mathbb{R}^2$, we let $\mathrm{ETSP}(\mathcal{Q})$ denote the length of the shortest closed path through all points in $\mathcal{Q}$. The following result characterizes the length of this path when $\mathcal{Q} \subset [0, \ell(n)]^2$.

Theorem 1 (ETSP tour length, [15]). *For every set of n points $\mathcal{Q} \subset [0, \ell(n)]^2$, we have $\mathrm{ETSP}(\mathcal{Q}) \in O(\sqrt{n}\ell(n))$.*

The problem of computing an optimal ETSP tour is known to be NP-complete. However, there exist many efficient approximation algorithms. For example, the *Christofides' algorithm* [16], computes a tour that is no longer than $3/2$ times the optimal in $O(n^3)$ computation time.

2.2 Bins and balls

Occupancy problems, or bins and balls problems, are concerned with randomly distributing m balls into n equally sized bins. The two results we present here will be useful in our analysis.

Theorem 2 (Bins and balls properties, [17,18]). *Consider uniformly randomly distributing m balls into n bins and let γ_n be any function such that $\gamma_n\theta_0 + \infty$ as $n\theta_0 + \infty$. The following statements hold:*

1. *if $m = n$, then w.h.p. each bin contains $O\left(\frac{\log n}{\log\log n}\right)$ balls;*
2. *if $m = n\log n + \gamma_n n$, then w.h.p. there are no empty bins, and each bin contains $O(\log n)$ balls;*
3. *if $m = n\log n - \gamma_n n$, then w.h.p. there exists an empty bin;*
4. *if $m = Kn\log n$, where $K > 1/\log(4/\mathrm{e})$, then w.h.p. every bin contains $\Theta(\log n)$ balls.*

We will be interested in dividing a square environment into equally sized and openly disjoint square bins, such that the side length $\ell(B)$, of each bin is small in some sense. To do this, we require the following simple fact.

Lemma 1 (Dividing the environment). *Given $n \in \mathbb{N}$ and $r > 0$, consider an environment $\mathcal{E}(n) := [0, \ell(n)]^2$. If $\mathcal{E}(n)$ is partitioned into b^2 equally sized and openly disjoint square bins, where*

$$b := \lceil \sqrt{5}\ell(n)/r \rceil, \tag{1}$$

then $\ell(B) \leq r/\sqrt{5}$. Moreover, if $x, y \in \mathcal{E}(n)$ are in the same bin or in adjacent bins, then $\|x - y\| \leq r$.

2.3 Random geometric graphs

For $n \in \mathbb{N}$ and $r \in \mathbb{R}_{>0}$, a planar *geometric graph* $G(n, r)$ consists of n vertices in $\mathbb{R}^2$, and undirected edges connecting all vertex pairs $\{x, y\}$ with $\|x - y\| \leq r$. If the vertices are randomly distributed in some subset of $\mathbb{R}^2$, we call the graph a *random geometric graph*.

Theorem 3 (Connectivity of random geometric graphs, [19]). *Consider the random geometric graph $G(n, r)$ obtained by uniformly randomly distributing n points in $[0, \ell(n)]^2$. If*

$$\pi \left(\frac{r}{\ell(n)} \right)^2 = \frac{\log n + c(n)}{n},$$

then $G(n, r)$ is connected w.h.p. if and only if $c(n)\theta_0 + \infty$ as $n\theta_0 + \infty$.

This theorem will be important for understanding some of our results. If we randomly deploy n agents with communication range $r > 0$ in an environment $[0, \ell(n)]^2$, then the communication graph is connected if $\ell(n) \leq r\sqrt{n/\log n}$.

3 Network model and problem statement

In this section we formalize our agent and target models and define the sparse and dense environments.

3.1 Robotic network model

Consider n agents in an environment $\mathcal{E}(n) := [0, \ell(n)]^2 \subset \mathbb{R}^2$, where $\ell(n) > 0$ (that is, $\mathcal{E}(n)$ is a square with side length $\ell(n)$). The environment $\mathcal{E}(n)$ is compact for each n but its size depends on n. A robotic agent, $\mathcal{A}^{[i]}$, $i \in \mathcal{I} := \{1, \ldots, n\}$, is described by the tuple

$$\mathcal{A}^{[i]} := \{\text{UID}^{[i]}, \mathbf{p}^{[i]}, r, \mathbf{u}^{[i]}, M^{[i]}\},$$

where the quantities are as follows: Its unique identifier (UID) is $\text{UID}^{[i]}$, taken from the set $I_{\text{UID}} \subset \mathbb{N}$. Note that, each agent does not know the set of UIDs being used and thus does not know the order. Its position is $\mathbf{p}^{[i]} \in \mathcal{E}(n)$. Its communication range is $r > 0$, i.e., two agents, $\mathcal{A}^{[i]}$ and $\mathcal{A}^{[k]}$, $i, k \in \mathcal{I}$, can communicate if and only

if $\|\mathbf{p}^{[i]} - \mathbf{p}^{[k]}\| \leq r$. Its continuous time velocity input is $\mathbf{u}^{[i]}$, corresponding to the kinematic model $\dot{\mathbf{p}}^{[i]} = \mathbf{u}^{[i]}$, where $\|\mathbf{u}^{[i]}\| \leq v_{\max}$ for some $v_{\max} > 0$. Finally, its memory is $M^{[i]}$ and is of size $|M^{[i]}|$. From now on, we simply refer to agent $\mathcal{A}^{[i]}$ as agent i. We assume the agents move in continuous time and communicate according to a discrete time schedule $\{t_k\}_{k\in\mathbb{N}}$. We assume $|t_{k+1} - t_k| \leq t_{\max}$, for all $k \in \mathbb{N}$, where $t_{\max} \in \mathbb{R}_{>0}$. At each communication round, agents can exchange messages of length $O(\log n)$.[1]

3.2 The target assignment problem

Let $\mathcal{Q} := \{\mathbf{q}_1, \ldots, \mathbf{q}_n\}$ be a set of distinct target locations, $\mathbf{q}_j \in \mathcal{E}(n)$ for each $j \in \mathcal{I}$. Agent i's memory, $M^{[i]}$, contains a copy of $\mathcal{Q}$, which we denote $\mathcal{Q}^{[i]}$. To store $\mathcal{Q}^{[i]}$ we must assume the size of each agents' memory, $|M^{[i]}|$, is in $\Omega(n)$. We refer to the assumption that each agent knows all target positions as the *full knowledge* assumption (for a more detailed discussion of this assumption see [12]). Our goal is to solve the *(full knowledge) target assignment problem*:

> Determine an algorithm for $n \in \mathbb{N}$ agents, with attributes as described above, satisfying the following requirement. There exists a time $T > 0$ such that for each target $\mathbf{q}_j \in \mathcal{Q}$, there is a unique agent $i \in \mathcal{I}$, with $\mathbf{p}^{[i]}(t) = \mathbf{q}_j$ for all $t \geq T$.

3.3 Sparse and dense environments

We wish to study the scalability of a particular approach to the target assignment problem; that is, how the completion time increases as we increase the number of agents, n. The velocity $v_{\max}$ and communication range r of each agent are independent of n. However, we assume that the size of the environment increases with n in order to accommodate an increase in agents. Borrowing terms from the random geometric graph literature [19], we say the environment is sparse if, as we increase the number of agents, the environment grows quickly enough that the density of agents (as measured by the sum of their communication footprints) decreases; we say the environment is critical, if the density is constant, and we say the environment is dense if the density increases. Formally, we have the following definition.

Definition 1 (Dense, critical and sparse environments). *The environment* $\mathcal{E}(n) := [0, \ell(n)]^2$ *is* sparse *if* $\ell(n)/\sqrt{n}\theta_0 + \infty$ *as* $n\theta_0 + \infty$, critical *if* $\ell(n)/\sqrt{n}\theta_0 C \in \mathbb{R}_{>0}$ *as* $n\theta_0 + \infty$, *and* dense *if* $\ell(n)/\sqrt{n}\theta_0 0$, *as* $n\theta_0 + \infty$.

It should be emphasized that a dense environment does not imply that the communication graph between agents is dense. On the contrary, from Theorem 3 we see that the communication graph at random agent positions in a dense environment may not even be connected.

[1] $\Omega(n)$ bits are required to represent an ID, unique among n agents.

4 Sparse environments

We begin by studying the case when the environment is sparse, and thus there is very little communication between agents. We introduce a natural approach to the problem in the form of a class of distributed algorithms, called *assignment-based motion*. We give a worst-case lower bound performance of the assignment-based motion class. Next, we introduce a control and communication algorithm, called ETSP ASSGMT. In this algorithm, each agent precomputes an optimal tour through the n targets, turning the cloud of target points into an ordered ring. Agents then move along the ring, looking for the next available target. When agents communicate, they exchange information on the next available target along the ring. We show that in sparse or critical environments, the ETSP ASSGMT algorithm is an asymptotically optimal among all algorithms in the assignment-based motion class.

4.1 Assignment-based algorithms with lower bound analysis

Here we introduce and analyze a class of deterministic algorithms for the target assignment problem. The assignment-based motion class can be described as follows.

Outline of assignment-based motion class

Initialization: In this class of algorithms agent i initially selects the closest target in $\mathcal{Q}^{[i]}$, and sets the variable $\text{curr}^{[i]}$ (agent i's current target), to the index of that target.

Motion: Agent i moves toward the target $\text{curr}^{[i]}$ at speed $v_{\max}$.

Communication: If agent i communicates with an agent k that is moving toward $\text{curr}^{[k]} = \text{curr}^{[i]}$, and if agent k is closer to $\text{curr}^{[i]}$ than agent i, then agent i "removes" $\text{curr}^{[i]}$ from $\mathcal{Q}^{[i]}$ and selects a new target.

For this class of algorithms it is convenient to adopt the following conventions: we say that agent $i \in \mathcal{I}$ is *assigned* to target $\mathbf{q}_j \in \mathcal{Q}$, when $\text{curr}^{[i]} = j$. We say that agent $i \in \mathcal{I}$ *enters a conflict* over the target $\text{curr}^{[i]}$, when agent i receives a message, $\text{msg}^{[k]}$, with $\text{curr}^{[i]} = \text{curr}^{[k]}$. Agent i *loses the conflict* if agent i is farther from $\text{curr}^{[i]}$ than agent k, and *wins the conflict* if agent i is closer to $\text{curr}^{[i]}$ than agent k, where ties are broken by comparing UIDs. Note that if an agent is assigned to the same target as another agent, it will enter a conflict in finite time.

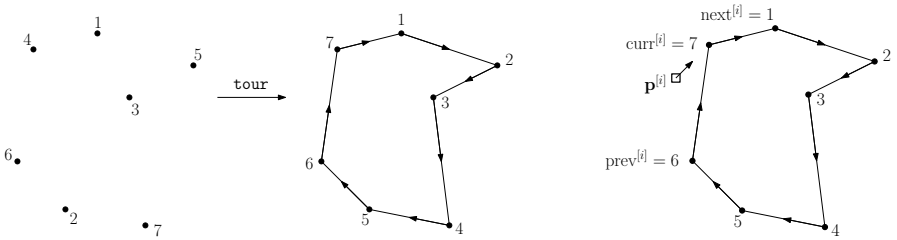
Theorem 4 (Time complexity lower bound for target assignment). *Consider n agents, with communication range $r > 0$, in an environment $[0, \ell(n)]^2$. If $\ell(n) \geq r\sqrt{n}$, then for all algorithms in the assignment-based motion class, the time complexity of the target assignment problem is in $\Omega(\sqrt{n}\ell(n))$.*

In other words, the target assignment time complexity is lower bounded when the environment grows faster than some critical value, that is, when the environment is sparse or critical.

4.2 The ETSP ASSGMT algorithm with upper bound analysis

In this section we introduce the ETSP ASSGMT algorithm—an algorithm within the assignment-based motion class. We will show that when the environment is sparse or critical, this algorithm is asymptotically optimal. In the following description of ETSP ASSGMT it will be convenient to assume that the target positions are stored in each agents memory as an array, rather than as an unordered set. That is, we replace the target set $\mathcal{Q}$ with the target n-tuple $\mathbf{q} := (\mathbf{q}_1, \ldots, \mathbf{q}_n)$, and the local target set $\mathcal{Q}^{[i]}$ with the n-tuple $\mathbf{q}^{[i]}$. The algorithm can be described as follows.

For each $i \in \mathcal{I}$, agent i computes a constant factor approximation of the optimal ETSP tour of the n targets in $\mathbf{q}^{[i]}$, denoted $\mathrm{tour}(\mathbf{q}^{[i]})$. We can think of tour as a map which reorders the indices of $\mathbf{q}^{[i]}$; $\mathrm{tour}(\mathbf{q}^{[i]}) = (\mathbf{q}^{[i]}_{\sigma(1)}, \ldots, \mathbf{q}^{[i]}_{\sigma(n)})$, where $\sigma : \mathcal{I}\theta_0\mathcal{I}$ is a bijection. This map is independent of i since all agents use the same method. An example is shown in Fig. 1(a). Agent i then replaces its n-tuple $\mathbf{q}^{[i]}$ with $\mathrm{tour}(\mathbf{q}^{[i]})$.



(a) The map tour orders the given targets (b) Initialization of agent i

Fig. 1. Initialization of ETSP ASSGMT

Next, agent i computes the index of the closest target in $\mathbf{q}^{[i]}$, and calls it $\mathrm{curr}^{[i]}$. Agent i also maintains the index of the next target in the tour which may be available, $\mathrm{next}^{[i]}$, and first target in the tour before $\mathrm{curr}^{[i]}$ which may be available, $\mathrm{prev}^{[i]}$. Thus, $\mathrm{next}^{[i]}$ is initialized to $\mathrm{curr}^{[i]} + 1 \pmod n$ and $\mathrm{prev}^{[i]}$ to $\mathrm{curr}^{[i]} - 1 \pmod n$. In order to "remove" assigned targets from the tuple $\mathbf{q}^{[i]}$, agent i also maintains the n-tuple, $\mathrm{status}^{[i]}$. Letting $\mathrm{status}^{[i]}(j)$ denote the jth entry in the n-tuple, the entries are given by

$$\mathrm{status}^{[i]}(j) = \begin{cases} 0, & \text{if agent } i \text{ knows } \mathbf{q}^{[i]}_j \text{ is assigned to another agent,} \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

Thus, $\mathrm{status}^{[i]}$ is initialized as the n-tuple $(1, \ldots, 1)$. The initialization is depicted in Fig. 1(b).

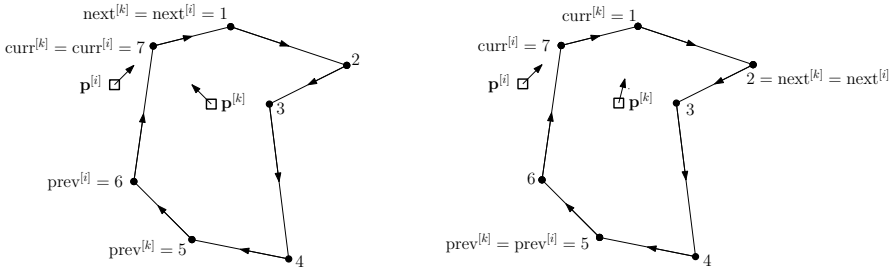
Finally, at each communication round agent i executes the algorithm COMM-RD described below.

Outline of COMM-RD algorithm for agent i

1: Broadcast msg$^{[i]}$, consisting of the targets, prev$^{[i]}$, curr$^{[i]}$, and next$^{[i]}$, the distance to the current target $d^{[i]}$, and UID$^{[i]}$.
2: **for all** messages, msg$^{[k]}$, received **do**
3: Set status$^{[i]}(j)$ to assigned ('0') for each target j from prev$^{[k]} + 1 \pmod n$ to next$^{[k]} - 1 \pmod n$ not equal to curr$^{[i]}$.
4: **if** prev$^{[k]}$ = next$^{[k]}$ = curr$^{[k]} \neq$ curr$^{[i]}$, **then** set the status of curr$^{[k]}$ to 0 because it was missed in the previous step.
5: **if** curr$^{[i]}$ = curr$^{[k]}$ but agent i is farther from curr$^{[i]}$ than agent k (ties broken with UIDs) **then**
6: Set the status of curr$^{[i]}$ to assigned ('0').
7: **if** curr$^{[i]}$ = curr$^{[k]}$ and agent i is closer than agent k **then**
8: Set the status of next$^{[i]}$ and next$^{[k]}$ to assigned ('0').
9: Update curr$^{[i]}$ to the next target in the tour with status available ('1'), next$^{[i]}$ to the next available target in the tour after curr$^{[i]}$, and prev$^{[i]}$ to the first available target in the tour before curr$^{[i]}$.

In summary, the ETSP ASSGMT algorithm is the triplet consisting of the initialization of each agent, the motion law (move toward curr$^{[i]}$ at speed $v_{\max}$), and the COMM-RD algorithm executed at each communication round.

Fig. 2 gives an example of COMM-RD resolving a conflict between agents i and k, over curr$^{[i]}$ = curr$^{[k]}$. The proposed algorithm enjoys plenty of useful properties,



(a) Setup prior to a conflict. (b) Setup after resolution of the conflict.

Fig. 2. The resolution of a conflict between agents i and k over target 7. Agent i wins the conflict since it is closer to target 7 than agent k.

which are valid for any communication graph which contains the geometric graph with parameter r as a subgraph. A complete discussion is contained in [12]. Based on a careful application of Theorem 1, one can derive the following key result.

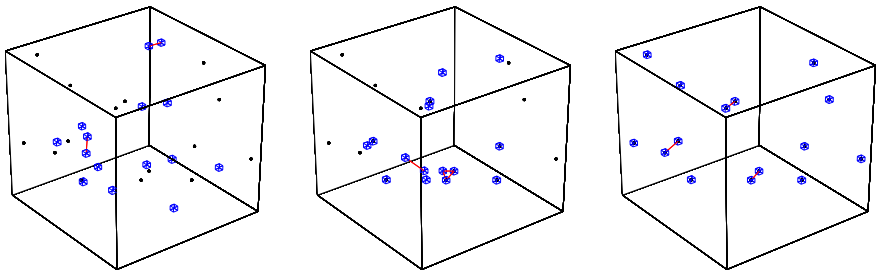
Theorem 5 (Correctness and time complexity for ETSP ASSGMT). *For any $n \in \mathbb{N}$,* ETSP ASSGMT *solves the target assignment problem. Furthermore, consider an environment $[0, \ell(n)]^2$. If $t_{\max} < r/v_{\max}$, then* ETSP ASSGMT *solves the target assignment problem in $O(\sqrt{n}\ell(n) + n)$ time. If, in addition, $\ell(n) > r\sqrt{n}$, then*

the time complexity is in $\Theta(\sqrt{n}\ell(n))$, and ETSP ASSGMT *is asymptotically optimal among algorithms in the assignment-based motion class.*

The above theorem gives a complexity bound for the case when r and $v_{\max}$ are fixed constants, and $\ell(n)$ grows with n. An equivalent setup is to consider ℓ fixed and allow the robots' attributes, r and $v_{\max}$, to vary inversely with the n, specifically, r and v proportional to $\sqrt{n}$.

Corollary 1 (Complexity with congestion). *Consider n agents moving with speed $\widetilde{v}_{\max}(n) = n^{-1/2}$ and communication radius $\widetilde{r}(n) = r_0 n^{-1/2}$, with $r_0 < 1$, in the environment $[0,1]^2$. Then* ETSP ASSGMT *solves the target assignment problem with time complexity in $\Theta(n)$.*

For simplicity we have presented our time complexity results in the planar environment $[0, \ell(n)^2]$. However, in [12] we derive bounds for the more general environment $[0, \ell(n)]^d$, $d \geq 1$. A simulation in $[0, 100]^3 \subset \mathbb{R}^3$ with $r = 15$ and $v = 1$ is shown in Fig. 3. To compute the ETSP tour we have used the `concorde` TSP solver.[2] The initial configuration shown in Fig. 3(a) consists of uniformly randomly generated target and agent positions.



(a) Initial configuration (b) Positions at time 30 (c) Complete assignment

Fig. 3. Simulation for 15 agents, $v_{\max} = 1$, $r = 15$ in $\mathcal{E} = [0, 100]^3$. The targets are spheres. The agents are cubes. An edge is drawn when two agents are communicating.

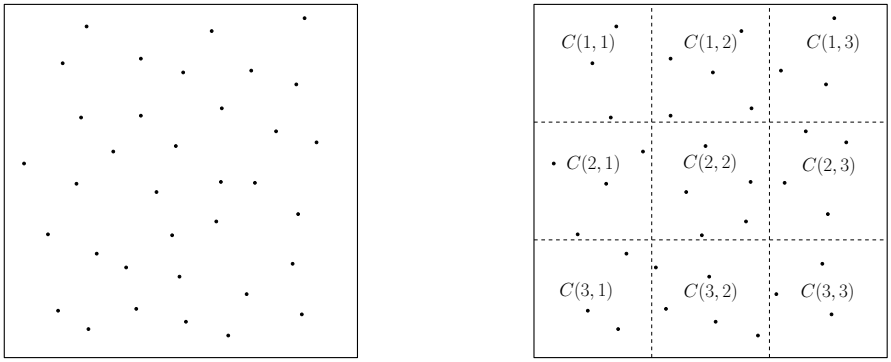
5 Dense environments

In the previous section we presented the ETSP ASSGMT algorithm which has provably good performance in sparse environments. In this section we introduce the GRID ASSGMT algorithm for dense environments in which communication is more prevalent. We will show that it has better worst-case performance than ETSP ASSGMT in dense environments, and that it possesses very good stochastic performance.

[2] The `concorde` TSP solver is available for research use at `http://www.tsp.gatech.edu/concorde/index.html`

5.1 The GRID ASSGMT algorithm with complexity analysis

In the GRID ASSGMT algorithm we assume that each agent knows the target positions, $\mathcal{Q}$, and the quantity $\ell(n)$ which describes the size of the environment. With this information, each agent partitions the environment into b^2 equally sized square cells, where $b \in \mathbb{N}$. It then labels the cells like entries in a matrix, so cell $C(w, c)$ resides in the wth row and cth column. This is shown in Fig. 4(b). Since the agents started with the same information, they all create the same partition.



(a) 35 targets in $\mathcal{E}(n)$. (b) $\mathcal{E}(n)$ divided into $b^2 = 9$ cells.

Fig. 4. Dividing the environment into 9 cells.

In light of Lemma 1, we see that when b is given by $\lceil \sqrt{5}\ell(n)/r \rceil$, as in equation (1), the communication graph between agents in a cell is complete, and communication between agents in adjacent cells is also possible. With this in mind, an outline of the GRID ASSGMT algorithm is as follows.

Outline of the GRID ASSGMT algorithm

Initialization: Each agent partitions the environment into b^2 equally sized square cells, where b is given in Lemma 1, and the cells are labeled as in Fig. 4(b).

All agents: In each cell, all agents in the cell find a maximum matching between agents and targets occupying the cell. Accordingly, agents are labeled assigned or unassigned.

Assigned agents: In each cell, all assigned agents elect a leader among them. All assigned agents, except the leaders, send their assignment information to their respective leader and then go silent.

Cell leaders: The leader in each cell communicates to leader in the cell directly above. As a result, each leader obtains an estimate of the number of available targets in all cells below it, in its column.

Unassigned agents: First, each unassigned agent seeks a free target in its column by entering cells and querying the corresponding leader.
Second, if all targets in the unassigned agent's column are assigned, then the agent moves to the top of its column and along the top row. The agent gathers from each leader in the top row the number of available targets in the leader's column. When the agent finds a column with available targets, it travels down that column to find the free target.

To implement this algorithm agent i maintains the following variables in its memory. The variable currcell$^{[i]}$ keeps track of the cell which agent i currently occupies. The set $\mathcal{Q}^{[i]}(w,c)$ which contains the targets in cell $C(w,c)$. The variable leader$^{[i]}$ which is set to $C(w,c)$ if agent i is the leader of $C(w,c)$, and `null` otherwise. The array colstatus$^{[i]}$, where colstatus$^{[i]}(c)$ is set to `full` if column c contains no available targets, and `notfull` if agent i thinks column c may contain an available target. The variable dircol$^{[i]} \in \{\texttt{down}, \texttt{up}\}$ which contains the direction of travel in a column and dirrow$^{[i]} \in \{\texttt{left}, \texttt{right}\}$ which contains the direction in the first row. Finally, the variable curr$^{[i]}$ which contains agent i's assigned target, or the entry `null`.

After initializing these variables, each agent runs an algorithm which allows the agents to compute a local maximum matching, and elect a leader, in each cell. Since the communication graph in each cell is complete, this can be done in one communication round by receiving the UIDs of each agent in the cell [13].

After the maximum matching and leader election the agents have been separated into three roles; assigned leader agents, assigned non-leader agents, and unassigned agents. The unassigned agents run an algorithm in which they try to find a free target. The leader of each cell runs an algorithm in which they update their estimates of available targets in various parts of the grid, and assigns unassigned targets in its cell. The leader of cell $C(w,c)$, agent i, maintains the following quantities to assign targets in its cell, and estimate the number of available targets in cells below. Agent i maintains: diff$^{[i]}(w,c)$, which records the difference between the number of targets and agents in cell $C(w,c)$; diffbelow$^{[i]}(w,c)$ which records agent i's estimate of the difference between the number of agents and targets in cells $C(w+1,c),\ldots,C(b,c)$; and taravail$^{[i]}(w,c)$ which contains the available targets in $C(w,c)$. Finally, if agent i is the leader of $C(1,c)$ in the first row, it maintains diffright$^{[i]}(c)$ which is agent i's estimate of the number of available targets in columns $c+1,\ldots,b$.

In summary, The GRID ASSGMT algorithm is the 4-tuple consisting of the initialization, the maximum matching and leader election algorithm, the unassigned agent algorithm, and the leader algorithm.

We can now state the main results on the GRID ASSGMT algorithm.

Theorem 6 (Correctness and worst-case upper bound). *For any initial positions of n targets and n agents in $[0,\ell(n)]^2$,* GRID ASSGMT *solves the target assignment problem in $O((\ell(n))^2)$ time.*

*Remark 1 (*GRID ASSGMT *vs.* ETSP ASSGMT*).* The worst-case bound for ETSP ASSGMT in Theorem 5 was $O(\sqrt{n}\ell(n))$. Thus, in sparse environments, when $\ell(n)$

grows faster than $\sqrt{n}$, ETSP ASSGMT performs better, and in dense environments GRID ASSGMT performs better. In critical environments, the bounds are equal. Thus, the two algorithms are complementary. In practice, if n, $\ell(n)$ and r are known, each robot in the network can determine which algorithm to run based on the following test: ETSP ASSGMT is run if $\ell(n)/\sqrt{n} > r$ and GRID ASSGMT is run if $\ell(n)/\sqrt{n} < r$. •

In the following theorem we will see that for randomly placed targets and agents, the performance of GRID ASSGMT is considerably better than in the worst-case. The proofs of the following theorems utilize the results on bins and balls problems in Section 2.

Theorem 7 (Stochastic time complexity). *Consider n agents and n targets, uniformly randomly distributed in $[0, \ell(n)]^2$. If $\ell(n) \leq r/\sqrt{5}\sqrt{(n/K \log n)}$, where $K > 1$, then* GRID ASSGMT *solves the target assignment problem in $O(\ell(n))$ time with high probability.*

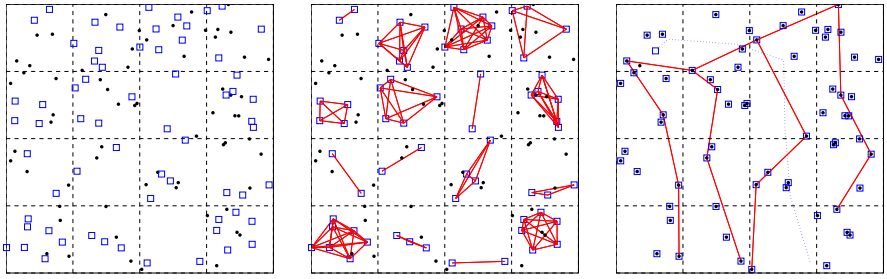
Remark 2 (Generalization of Theorem 7). The bound in Theorem 7 holds for any initial positions such that every cell contains at least one target and at least one agent. •

Theorem 8 (Stochastic time complexity: More agents than targets). *Consider n agents and $n/\log n$ targets, uniformly randomly distributed in $[0, \ell(n)]^2$. If $\ell(n) \leq r/\sqrt{5}\sqrt{(n/K \log n)}$, where $K > 1/\log(4/\mathrm{e})$, then w.h.p.,* GRID ASSGMT *solves the target assignment problem in $O(1)$ time.*

A representative of simulation of GRID ASSGMT for 65 agents and targets uniformly randomly distributed in a dense environment is shown in Fig. 5(a)–(c). In Fig. 5(c) a dashed blue trail shows the trajectory for the final agent as it is about to reach its target in cell $C(1, 1)$. Fig. 5.1 contains a Monte Carlo simulation for uniformly randomly generated agents and targets. The side length $\ell(n)$ satisfies the bound in Theorem 7, and the agents move at unit speed. Each data point is the mean completion time of 30 trials, where each trial was performed at randomly generated agent and target positions. Error bars show plus/minus one standard deviation. The mean completion time lies between $2\ell(n)$ and $3\ell(n)$. This agrees with the $O(\ell(n))$ bound in Theorem 7 and gives some idea as to the constant in front of this bound.

5.2 A sensor based version

In describing the GRID ASSGMT algorithm, we assumed that each agent knows the position of all targets. The algorithm also works when each agent does not know the position of any targets, but has a sensing range r_{sense}, with which it can sense the positions of targets in range. If each agent can partition the environment as in Fig. 4, and if $r_{\text{sense}} \geq \sqrt{2/5}r$ so that each agent can sense the position of all targets in its current cell, then GRID ASSGMT (with minor modifications) solves the target assignment problem, and the completion time results still hold.



(a) Initial agent and target positions, and grid.

(b) Maximum assignment and leader election.

(c) Final agent reaching target.

Fig. 5. A simulation of 65 agents in a dense environment. Targets are black disks and agents are blue squares. Red lines are drawn when two agents are communicating.

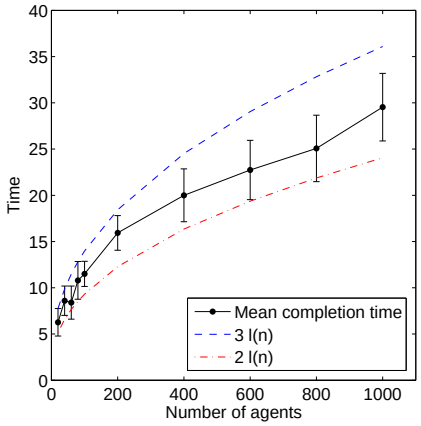


Fig. 6. A Monte Carlo simulation. Each data point is the mean of 30 trials.

5.3 Congestion issues

Since wireless communication is a shared medium, simultaneous messages sent in close proximity will collide, resulting in dropped packets. In fact, clear reception of a signal requires that no other signals are present at the same point in time and space. As the density of agents increases (as measured by their communication footprints), so does wireless communication congestion. Thus, in dense environments, one would ideally account for the effects of congestion. In the design of GRID ASSGMT we have tried to limit the amount of simultaneous communication. To this end we introduced a leader in each cell, who sent messages (of size $O(\log n)$) only to its adjacent cells, and all other assigned agents were silent. However, to fully take wireless congestion into account, we would require a more sophisticated communication model than the geometric graph.

6 Conclusion and extensions

In this chapter we have discussed two complementary algorithms for the target assignment problem, ETSP ASSGMT and GRID ASSGMT. We have shown that ETSP ASSGMT has better performance in sparse environments, where as, GRID ASSGMT has better performance in dense environments. There are many future research directions such as extensions to vehicles with motion constraints, or to the case when targets are dynamically appearing and disappearing. Another area of future research is to develop a communication framework which adequately models congestion and media access problems that are inherently present in wireless communications.

Acknowledgments: In Giorgio's honor

The second author dedicates this work to Giorgio Picci. I am honored to have had him as my Laurea advisor where he introduced me to the exciting world of scientific research. His passion for control theory, applied mathematics and geometry was highly contagious and continues to be a part of my life today. I am honored to be able to dedicate this work to a man I consider an inspiration. Grazie di cuore!

References

1. B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*. New York: Springer Verlag, 3 ed., 2005.
2. J. E. Hopcroft and R. M. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM Journal on Computing*, vol. 2, no. 4, pp. 225–231, 1973.
3. H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics*, vol. 2, pp. 83–97, 1955.
4. R. Burkard, "Selected topics on assignment problems," *Discrete Applied Mathematics*, vol. 123, pp. 257–302, 2002.
5. D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA: Athena Scientific, 1997.
6. M. F. Godwin, S. Spry, and J. K. Hedrick, "Distributed collaboration with limited communication using mission state estimates," in *American Control Conference*, (Minneapolis, MN), pp. 2040–2046, June 2006.
7. M. Alighanbari and J. P. How, "Robust decentralized task assignment for cooperative UAVs," in *AIAA Conf. on Guidance, Navigation and Control*, (Keystone, CO), Aug. 2006.
8. 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*, (Denver, CO), pp. 3472–3477, 2003.
9. B. J. Moore and K. M. Passino, "Distributed task assignment for mobile agents," *IEEE Transactions on Automatic Control*, 2006. to appear.
10. M. Zavlanos and G. Pappas, "Dynamic assignment in distributed motion planning with local information," in *American Control Conference*, (New York), July 2007. To appear.
11. G. Arslan and J. S. Shamma, "Autonomous vehicle-target assignment: a game theoretic formulation," *IEEE Transactions on Automatic Control*, Feb. 2006. Submitted.

12. S. L. Smith and F. Bullo, "Target assignment for robotic networks: Asymptotic performance under limited communication," in *American Control Conference*, (New York), July 2007. To appear.
13. S. L. Smith and F. Bullo, "Target assignment for robotic networks: Worst-case and stochastic performance in dense environments," in *IEEE Conf. on Decision and Control*, (New Orleans, LA), Dec. 2007. Submitted.
14. D. A. Castañón and C. Wu, "Distributed algorithms for dynamic reassignment," in *IEEE Conf. on Decision and Control*, (Maui, HI), pp. 13–18, Dec. 2003.
15. K. J. Supowit, E. M. Reingold, and D. A. Plaisted, "The traveling salesman problem and minimum mathcing in the unit square," *SIAM Journal on Computing*, vol. 12, pp. 144–156, 1983.
16. N. Christofides, "Worst-case analysis of a new heuristic for the traveling salesman problem," Tech. Rep. 388, Carnegie-Mellon University, Apr. 1976.
17. R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge, UK: Cambridge University Press, 1995.
18. F. Xue and P. R. Kumar, "The number of neighbors needed for connectivity of wireless networks," *Wireless Networks*, vol. 10, no. 2, pp. 169–181, 2004.
19. M. Penrose, *Random Geometric Graphs*. Oxford Studies in Probability, Oxford, UK: Oxford University Press, 2003.