Swarms Can be Rational*

Yinon Douchan
Tel Aviv University
Tel Aviv, Israel
yinondouchan@mail.tau.ac.il

Ran Wolf
Bar Ilan University
Ramat Gan, Israel
ran.wolf@outlook.com

Gal A. Kaminka
The MAVERICK Group, Bar Ilan University
Ramat Gan, Israel
galk@cs.biu.ac.il

ABSTRACT

A fundamental challenge in multi-robot systems is spatial coordination (avoiding collisions) between robots, each under its own control. Swarm methods, where by robots coordinate *ad-hoc* and *locally*, offer a promising approach. However, while empirically demonstrated to be viable in practice, no guarantees of performance are known. This paper formalizes a class of multi-robot cooperative tasks as potential extensive-form games. We show that the system coordination overhead is a potential function, forming a connection between the theoretical maximum-payoff equilibrium of the system, and the rational self-interested choices of individual robots during task execution: *robot swarms can be rational in theory*. We then show how to approximate the rational decision-making in practice using reinforcement learning, using internal measures for rewards. We empirically show this leads to consistent optimal performance in with physical and simulated robots.

KEYWORDS

Multi-Robot Systems; Swarm; Game-Theory

ACM Reference Format:
Yinon Douchan, Ran Wolf, and Gal A. Kaminka. 2019. Swarms Can be Rational. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

Distributed multi-robot systems are comprised of multiple robots, each under its own control. Typically, the robots carry out a tasks towards a global goal. Examples include *coverage* [1, 6, 14, 22, 35], *patrolling* [2, 12, 25, 34], *foraging* [4, 11, 16, 18, 23, 24], *order picking* [17, 32] (made famous by Kiva Systems/Amazon Robotics), and more.

A fundamental challenge here is *spatial coordination*, i.e., avoiding and resolving collisions with others. Such coordination necessarily introduces overhead, and therefore both supports and competes with achievement of the goals of the robots. Centralized solutions to managing spatial coordination are computationally intractable [36], and are often not tolerant to mechanical and environment unpredictability (e.g., humans in the work area) [32]. Distributed approaches which rely on joint decision-making by the robots (e.g., [19, 33] require high communication availability and the capability of robots to assess not just their own state, but those of others. Thus a promising line of research explores swarm methods, where by robots coordinate *ad-hoc* and *locally*, with little or no communications [4, 18, 23, 24]. However, while empirically demonstrated to be a viable solution in practice, no theoretical guarantees of performance are known, nor a formalization of the task in a way that admits analysis.

This paper formalizes a class of multi-robot cooperative tasks as *potential extensive-form games*. The game itself requires knowledge (e.g., of the payoffs of others) that individual swarm robots do not have access to. However, we show that the system coordination overhead is a potential function for the game, forming a connection between the theoretical maximum-payoff equilibrium of the system, and the rational choices of individual robots during task execution. In other words, individual robots can make *rational coordination decisions* resulting in globally optimal equilibrium.

A swarm acting as prescribed is rational in theory. We show how to closely approximate the optimal decision-making in practice, relying on reinforcement learning to improve decision-making, while operating strictly within the limited capabilities of simple swarm robots, in both perception and computation. These approximations generalize and improve on earlier work, which emerges as a special case. We empirically evaluate the efficacy of these methods in two multi-robot domains: physical robots carrying out a variant foraging task, and simulated robots carrying out order-picking.

2 MOTIVATION AND BACKGROUND

To position our work in context, Figure 1 shows a perspective (previously described in [18]) on the structure of task execution in time, for an individual robot. The robot begins by executing its task, stopping when a spatial conflict occurs (e.g., a collision is imminent). It then selects a collision-handling method, which executes for a time. When the collision is averted, the robot can switch back to carrying out its task until another collision is imminent. This continues up until task execution has ended. The time interval between collisions is split into two [18], termed the *active time* (spent by the robot actively coordinating—shown in grey), and the *passive time* (no need to coordinate; the robot focuses on its task).

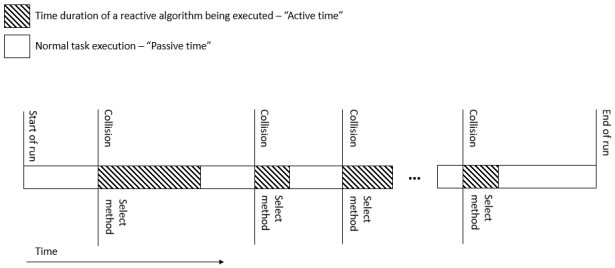

Figure 1: A Single Robot's timeline.

*This research was supported in part by ISF Smart Swarms Center, Grant #2306/18. As always, thanks to K. Ushi.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

This view of the robot's timeline allows us to position our work with respect to others. First, we note that collisions involve more than one robot. If all the robots are involved in each others' task execution and decision-making, the single-robot timeline above is also the multi-robot system timeline. Theoretically, multi-robot path planning reduces or eliminates the number of collisions, and explicit joint decision making can reduce their collective duration. However, the computational, communications, and actual cost of these methods can often be too high in practice, and thus reactive approaches are often employed. These follow two general directions: specific collision-handling algorithms which reduce the active time duration, and algorithm-selection methods which allow the robot to tailor its selection of collision-handling algorithm to the specific collision settings.

Collision Handling Algorithms. Purely reactive coordination algorithms are *myopic*. They respond to a collision with no consideration of future collisions. They are extremely simple to implement and use (both in practice and theory), and are generally task-independent (because they do not use information about the goals of the task). We use several such reactive algorithms in our work.

Some algorithms resolve collisions by moving backwards from the collision position for a fixed time or distance [23], potentially adding noise to movement [5]. Others try to keep one robot moving forward: the *aggression* family of algorithms allows the robot with the highest *aggression factor* gets the right of way [28].

Others carry out limited planning. We use a variant of *dynamic window* [13] in the experiments, as it makes assumptions that are met in practice. Other potential choices include Reciprocal Velocity Obstacles methods such as ORCA [27], or safe-navigation methods (e.g., [7]). However, they prove too complex for our simple robots.

Algorithm Selection. It is now understood that while each method is effective in some settings, no method is always effective [23, 24]: their performance changes with density and other factors. Thus a promising approach is to have the robots dynamically select between collision-handling algorithms.

Wolpert and Tumer [30] described the COIN framework, which models multi-agent systems where agents work to maximize global utility, but with little or no communications between them. They show that if agents can estimate the *wonderful life utility*—how the agent's actions (or lack thereof) impact global utility—then it is possible to use reinforcement learning to improve global utility in a guaranteed manner, in various domains including robotics [26, 29]. However, this relies on knowing the global utility, and/or the value (payoff) of others' actions. In practice, this is often not possible. We build on their work by showing how to approximate the wonderful life utility in practice in multi-robot settings

Erusalimchik et al. [18] propose a reward function (the coordination overhead of an interval—the ratio of the individual active time to the total interval duration), as the basis for preferring algorithms, using a stateless Q-learning variant [10]. We show later that theirs is a special case emerging from our work.

ALAN [15] is a method using reinforcement learning techniques with ORCA to improve global measures. It improves on either using only ORCA or only reinforcement learning. However, ALAN can only choose between alternatives within ORCA, and does not provide guarantees on performance, as we do here.

3 SWARMS AND GAME THEORY

We begin in Section 3.1 by introducing an abstract game-theoretic model of multi-robot tasks carried out by a swarm of robots. We then make incremental modifications to this abstract model, to bring it closer to the reality of physical interacting robots, when the robots cannot communicate (Sections 3.2–3.3). Finally, in section 3.4, we address the challenge of learning optimal actions according to the game-theoretic model we introduced.

3.1 Swarm tasks as extensive-form games

When considering the task multiple robots (each engaging in its own coordination method arbitration), we follow Erusalimchik et al. [18] in representing the task as an extensive form game between n robots. The extensive form game represents every possible outcome as a function of the sequence of parallel coordination actions taken by all robots in every collision during the run. In this context, the outcome is the utility of each of the robots in the alloted time.

The root node of the game tree represents the first collision. Given that there are n robots, the first n layers of the game tree will each represent a robot and its possible actions in the first collision. This is because we focus on non-communicating coordination methods, and thus we will treat each collision as having no information on the actions and utilities selected and gained by other robots.

The actions independently taken by players are coordination methods: The gains (payoffs) from taking them and the costs which they entail differ between robots and between collisions, but are theoretically accounted for. Each action takes time.

The next n layers will represent the second collision in the same manner. This sequence continues until a terminal node is reached—when the time for the task is done: A terminal node represents the end of the game (task) and holds the utility of each player. Since different actions can yield different time intervals between collisions, terminal nodes can each be of different depth depending on the sequence of collisions (and associated joint actions chosen) during the game. Each such sequence is represented as a path in the game tree. Each terminal node will hold a vector of numerical values representing the utilities of each robot in the system.

3.2 Folding the game into normal-form

The extensive form model of a task run represents every possible outcome of the task run. This is only of theoretical value, as no robot—nor their designers—can predict the outcome of future collisions, nor their timing, nor their impact on global payoffs. In reality, robots only know their history of previous collisions, and the immediately imminent collision. Indeed, in swarm settings robots cannot know of the other robots' choices (which theoretically affects their own) and thus even this information is hidden from them.

In order for robots to make decisions based only the history and current collision, we must draw a connection between the global final utility (payoff) theoretically reached using the extensive-form game, and the sequence of collisions in which the robots make collision-resolution choices. Robots may then rely on signals that are obtained during a joint collision.

To do this, we take an intermediate step and show how the extensive form game can be expressed as a sequence of normal

form games, each representing a single joint collision. We define the following:

- s_i^j: robot i's action at the j'th collision. s^j: joint action at collision j.
- $h_i^j = (s_i^1, s_i^2, ..., s_i^j)$: Robot i's history of actions until the j'th collision inclusive. History of all robots' actions until j inclusive: h^j.
- The *cost* of robot i at the j'th collision will be denoted as c_i^j.
- The *gain* of robot i at the j'th collision will be denoted as g_i^j.
- $u_i^j = g_i^j - c_i^j$: The *utility* of robot i at the j'th collision.
- U: The *global utility* of all the robots during the whole run.

We start with the most general case where outcomes of a robot at the j'th collision may depend on the entire history of play of all the robots up until collision j inclusive. This means that u_i^j, g_i^j, c_i^j are all functions of h^j. U will now depend on the entire history of play. Given that the number of collisions for the whole task run is c, U will be a function of h^c and will be defined as the sum of utilities of every robot and every collision during the task run (Eq. 1).

$$U(h^c) = \sum_{i \in N} \sum_{j=1}^{c} u_i(h^j) = \sum_{i \in N} \sum_{j=1}^{c} (g_i(h^j) - c_i(h^j)) \tag{1}$$

We can look at each joint collision as a normal-form (matrix) game representing the outcomes of this collision only, rather than the whole task run. For the j'th collision, the player set of this matrix is the set of robots and the action set of each robot is its set of coordination methods. Given the history of joint actions played up until collision j (inclusive), h^j, the payoffs of this matrix will be the sum of the utilities of the robots obtained only for the j'th collision $\sum_{i \in N} u_i(h^j)$ as a function of the history of play. We call this matrix the *Folded Game Matrix*.

3.3 Global utility and folded matrices

Robots in a system have limited sensing and communication capabilities. They are unable to know the utilities of other robots, even in the same joint action. Indeed, each robot does not even know how its own action affects its own immediate utility. The only information available to it is data from its own sensors and internal state information. We formally tie the active and passive times of the collision to the utility of the robot resulting from the collision. To do this we assume that individual gains in active time are zero (since a robot in active time is handling a collision), and therefore gains occur only in passive time: $g_i(h^j) = g_i(P_i(h^j))$. We will further assume that the gains are proportional to the passive time given history of play h^j: $g_i(h^j) = \alpha P_i(h^j)$, $\alpha = const$. We also assume that costs are constant, $c_i(h^j) = \beta(A_i(h^j) + P_i(h^j))$ where $\beta = const$.

3.3.1 Global Utility and Coordination Overhead.

Definition 3.1. The *Coordination Overhead (CO)* is the total amount of time the system was in active time divided by the total time invested in the task run: $CO(h^c) = \frac{1}{T} \sum_{i \in N} \sum_{j=1}^{c} A_i(h^j)$.

Since T is the sum of all cycle length of any of the robots' task run, we can write $T = \sum_{j=1}^{c} (A_i(h^j) + P_i(h^j))$ for any robot i. Therefore, CO can also be written as $CO(h^c) = \sum_{i \in N} \frac{\sum_{j=1} A_i(h^j)}{\sum_{j=1}^{c} (A_i(h^j) + P_i(h^j))}$. We will now show, given the above assumptions, that U is a linear decreasing function of CO, i.e., minimizing CO is maximizing U.

Theorem 3.2. *Given the assumptions on the cost and gain, U is a linear decreasing function of CO.*

Proof. $U = \sum_{i \in N} \sum_{j=1}^{c} [u_i(h^j)] = \sum_{i \in N} \sum_{j=1}^{c} [g_i(h^j) - c_i(h^j)] = \sum_{i \in N} \sum_{j=1}^{c} [\alpha P_i(h^j) - \beta(A_i(h^j) + P_i(h^j))] = \sum_{i \in N} \sum_{j=1}^{c} (\alpha P_i(h^j)) - \sum_{i \in N} \sum_{j=1}^{c} \beta(A_i(h^j) + P_i(h^j)) = T \frac{\alpha \sum_{i \in N} \sum_{j=1}^{c} P_i(h^j)}{T} - nT\beta = T\alpha(1 - CO(h^c)) - nT\beta = -T\alpha \cdot CO(h^c) + T(\alpha - n\beta)$ □

3.3.2 Coordination Overhead and the Folded Matrices.

We further assume that for every collision, the outcomes of the robots' method selection depend only on the current joint action performed and not on the history of all joint actions performed. This also means that no matter what the collision index is, as long as the joint action stays the same, the outcomes of this collision stay the same. Therefore, variables that depend on the history of joint actions played until collision j, $h^j \in S^j$, depend only on the joint action that was played in time j, $s^j \in S$. We can now denote the active time as $A_i(s^j)$ and since it does not vary in time, we can denote it as $A_i(s)$. The same applies for P_i, g_i, c_i, u_i and U. One consequence of this assumption is that instead of the task run being a sequence of different folded game martices depending on the history of play, it is now a single folded game matrix which is the same for every collision in the task run.

Given a joint action s and a robot i, we will define $EI_{tot}(s)$ to be the sum of the effectiveness indices of all robots: $EI_{tot}(s) = \sum_{i \in N} EI_i(s) = \sum_{i \in N} \frac{A_i(s)}{A_i(s)+P_i(s)}$. Let s^* be the joint action that minimizes EI_{tot}: $s^* = argmin_s(EI_{tot}(s))$. If the system always plays joint action s^* its CO will be: $CO(h^*) = \sum_{i \in N} \frac{c \cdot A_i(s^*)}{c \cdot [A_i(s^*)+P_i(s^*)]} = \sum_{i \in N} \frac{A_i(s^*)}{A_i(s^*)+P_i(s^*)} = EI_{tot}(s^*)$ where $h^* = (s^*, s^*, ..., s^*)$. We will now show that for every sequence of joint actions CO will be greater or equal to $EI_{tot}(s^*)$. This means that in order to minimize CO the system always needs to select s^* as the joint action.

Theorem 3.3. *for any number of collisions c and histories of play h^c, $CO(h^c) \geq EI_{tot}(s^*)$.*

Proof.

$$\begin{aligned}
CO(h^c) &= \sum_{i \in N} \frac{\sum_{j=1}^{c} A_i(s^j)}{\sum_{j=1}^{c} (A_i(s^j) + P_i(s^j))} & &= \sum_{i \in N} \frac{\sum_{j=1}^{c} A_i(s^j)}{T} \\
&= \frac{1}{T} \sum_{i \in N} \sum_{j=1}^{c} A_i(s^j) & &= \frac{1}{T} \sum_{j=1}^{c} \sum_{i \in N} A_i(s^j) \\
&= \frac{1}{T} \sum_{j=1}^{c} l(s^j) \sum_{i \in N} \frac{A_i(s^j)}{l(s^j)} & &= \frac{1}{T} \sum_{j=1}^{c} l(s^j) EI_{tot}(s^j) \\
&\geq \frac{1}{T} \sum_{j=1}^{c} l(s^j) EI_{tot}(s^*) & &= EI_{tot}(s^*) \frac{1}{T} \sum_{j=1}^{c} l(s^j) \\
&= EI_{tot}(s^*) \frac{1}{T} T & &= EI_{tot}(s^*)
\end{aligned}$$

□

However, robots only know internal properties; they cannot know s^*, which requires knowledge of the actions of other robots. We need to find a way to make the robots converge to s^* by using internal measurements only.

3.4 Learning Optimal Actions

3.4.1 Potential games. We show that the use of Potential Games [21] can address this challenge, A potential game is a normal form game where for every player i, the difference in the payoff of every unilateral deviation of player i's action s_i is related to the difference of a single potential function $\psi(s)$ mapping each joint action to a scalar. $\psi(s)$ can be seen as a global signal (not necessarily visible to the players) which depends on the joint action.

Potential games hold several important characteristics: First, they always have at least one pure-strategy Nash equilibrium. Also, when players use pure strategies, a change in one player's individual payoff due to changing its individual action will be aligned with the potential function. This means that in any potential game if one player chooses to change its action to an action with better payoff, the potential function will always benefit, and vice versa. Therefore, if players choose to maximize their individual payoffs, the system will converge to a pure-strategy Nash equilibrium, which would be at least a local optimum of the potential function.

In our case, we need to find a payoff function for each robot, such that the robots play a potential game with potential function EI_{tot}. Doing so will make the robots converge to an optimal joint action in terms of EI_{tot}.

3.4.2 Total EI as a Potential Function. In Section 3.3 we've seen that the problem of optimizing the global utility narrows down to minimizing EI_{tot} by converging to a single joint action s^* while still using only internal measurements.

In order to do so we use the *Wonderful Life Utility (WLU)* [31]. Given a global utility U, the WLU for robot i is a measurement of the difference between the resulting U and the global utility when robot i is absent. In terms of game theory, the absence of robot i is equivalent to the robot choosing a *null* action denoted by ϕ_i.

Definition 3.4. Wonderful Life Utility. Given a global utility U and a joint action $s = (s_i, s_{-i})$, the wonderful life utility of robot i is:

$$WLU_i^U(s_i, s_{-i}) = U(s_i, s_{-i}) - U(\phi_i, s_{-i})$$

The WLU is, in fact, a measurement of robot i's marginal contribution to U. It is known that agents that learn with WLU as a utility function play a potential game with the global utility as the potential function [3, 20].

Since our goal is to optimize EI_{tot}, we can now make the robots choose actions according the WLU of EI_{tot}. If robots do so, not only they will converge to a joint action, the potential function will be $\psi = EI_{tot}$. Therefore, this joint action will at least be a local minimum of EI_{tot} due to the properties of potential games. We will now show a closed expression of $WLU_i^{EI_{tot}}$.

Theorem 3.5. *let $l_i(s)$ be the cycle length of robot i given a joint action s: $l_i(s) = A_i(s) + P_i(s)$. The WLU of EI_{tot} for robot i takes the form $\frac{A_i(s)+A_i^{\phi}(s)}{A_i(s)+P_i(s)} - \frac{A_i(\phi_i,s_{-i})}{l_i(\phi_i,s_{-i})}$ where $A_i^{\phi}(s) = \sum_{j\in N\setminus\{i\}}(A_j(s)\frac{l_i(s)}{l_j(s)} - A_j(\phi_i,s_{-i})\frac{l_i(s)}{l_j(\phi_i,s_{-i})})$.*

Proof (sketch).

$$\begin{aligned} WLU_i^{EI_{tot}}(s) &= EI_{tot}(s) - EI_{tot}(\phi_i, s_{-i}) \\ &= \frac{A_i(s) + A_i^{\phi}(s)}{A_i(s) + P_i(s)} - \frac{A_i(\phi_i, s_{-i})}{l_i(\phi_i, s_{-i})} \end{aligned}$$

□

This is the most general form of the WLU of EI_{tot}. Different assumptions result in special cases that can be useful.

First, when a robot is absent from the system it cannot contribute to the system and its costs are zero. The expression $\frac{A_i(\phi_i,s_{-i})}{l_i(\phi_i,s_{-i})}$ is the effectiveness index of robot i when it is absent. Since we assumed that gains in active time are zero we can express the absence of the robot i as if it was always in active time during the cycle length $l_i(\phi_i, s_{-i})$. This assumption can be expressed as $A_i(\phi_i, s_{-i}) = l_i(\phi_i, s_{-i})$. Therefore we can see that $\frac{A_i(\phi_i,s_{-i})}{l_i(\phi_i,s_{-i})} = 1$. The WLU will now take the form $WLU_i^{EI_{tot}}(s) = \frac{A_i(s)+A_i^{\phi}(s)}{A_i(s)+P_i(s)} - 1$.

Also, collisions are synchronous in our model. This means that cycle length depends only on the joint action selected and not on the robot. For all pairs of robots $i, j \in N$ and all joint actions s: $l_i(s) = l_j(s)$. Therefore, we can remove the subscript and write $l(s)$. The effect of this assumption is that now $A_i^{\phi}(s)$ takes a simpler form: $A_i^{\phi}(s) = \sum_{j\in N\setminus\{i\}}(A_j(s) - A_j(\phi_i, s_{-i})\frac{l(s)}{l(\phi_i,s_{-i})})$

Finally, the absence of robot i does not affect cycle length: $l(s) = l(\phi_i, s_{-i})$. $A_i^{\phi}(s)$ can now be expressed as $A_i^{\phi}(s) = \sum_{j\in N\setminus\{i\}}(A_j(s) - A_j(\phi_i, s_{-i}))$. Now A_i^{ϕ} has a meaning - it is the change in the total active time of the system due to the absence of robot i.

Thus WLU takes the form

$$WLU_i^{EI_{tot}}(s) = \frac{A_i(s) + \sum_{j\in N\setminus\{i\}}(A_j(s) - A_j(\phi_i, s_{-i}))}{A_i(s) + P_i(s)} - 1$$

which can be written as

$$WLU_i^{EI_{tot}}(s) = \frac{A_i(s) + A_i^{\phi}(s)}{A_i(s) + P_i(s)} - 1$$

Omitting the -1 element will not change the optimization problem. From the above formula of WLU we can see that either minimizing $A_i(s)$ or $A_i^{\phi}(s)$ minimizes WLU. Comparing WLU to EI we can see that EI [18] a special case of WLU but with $A_i^{\phi} = 0$ (Henceforth, unless stated otherwise, we denote WLU of EI_{tot} as $WLU_i(s)$.

The WLU may now be used by robots as a reward function in a reinforcement learning algorithm. In practice, they will use WLU approximation $\widehat{WLU}$ (see Section 4). To do this, they keep track of A and P of intervals between collisions.

Assumption	Motivation
Gains in active time are zero	Robots cannot directly contribute to the task when they focus on conflict resolution and avoid collisions.
Gains are proportional to passive time	The more a robot works uninterrupted, the higher its gains will be. Assumption for theoretical derivation.
Costs are proportional to time	Robots spend resources (e.g., power) when operating. Longer operations leads to more spending. Assumption for theoretical derivation.
Outcomes of robots' actions do not depend on history of method choices	Without learning, the success of collision avoidance in past collisions does not impact its success in the current collision. For theoretical derivation.
When a robot is absent it cannot contribute to the system: its gains and costs are zero	By definition.
Cycle length is the same for all robots for a joint action	The theoretical model is synchronous.
The absence of a robot does not affect cycle length	The effects of a single robot in a swarm tend to be negligible. Assumption for theoretical derivation.

Table 1: Assumptions made in the development of the theoretical model, and the motivation for making them.

3.5 Summary of Assumptions

We summarize the assumptions made in the development of the model in Table 1.

4 LEARNING IN PRACTICE

There are gaps between the theory presented and the limitations in practice: (i) the WLU of EI_{tot} for a robot needs knowledge of other robots' measurements; (ii) collisions are not mutual—when robots are about to collide, there is no guarantee that either one of them will actually recognize a collision and act to avoid it and finally (iii) active and passive times (A, P) vary for the same method. These gaps are addressed below. There is another gap which is explained but not addressed in this paper (iv) - different clusters of robots can collide at the same time in different places.

4.1 Approximating WLU of total EI

WLU_i is composed of three elements: A_i, P_i and A_i^ϕ. A_i and P_i are internal to the robot, thus known. However, A_i^ϕ requires the robot to know the active times of other robots. This is impractical, as the effects of a robot on other robots often impossible to perceive accurately by swarm robots.

Thus, robots estimate $\widehat{WLU}_i$. For lack of space, we report here only on one such approximation, $\widehat{WLU_i(s)} = \frac{A_i(s)+n_a \cdot A_0}{A_i(s)+P_i(s)}$ where n_a is the number of robots affected by this robot (i.e., entered collision avoidance because of the robot), and A_0 is an estimate of how much each robot was affected in terms of active time added to it.

We focus here only on the two most useful approximators out of several we tested, as evident in the experiments. One is where $A_0 = 0$, i.e., the active times of other robots is assumed unaffected. This approximation is the special case EI_i [18]. The other approximation is *Minimum over Actions*, where the addition in active time to other robots is by finding the individual action $s' \in S_i$ that has the lowest average active time, i.e., $A_0 = \min_{s' \in S_i}(\frac{1}{|C(s')|}\sum_{j \in C(s')} A_i(s^j))$. Here, we define $C(s) \subseteq \{1, ..., c\}$ as the subset of collision indices where joint action s was played. In the same manner we will define $C(s_i)$ as the subset of collision indices where robot i chose individual action $s_i \in S_i$, regardless of the actions chosen by other robots.

4.2 Dealing with Non-Mutual Collisions

Collisions are asynchronous and often non-mutual. This means that even if robots collide at the same time, there is no guarantee that all of them will recognize they collided. This is mainly caused by limited sensing capabilities. For example, if a robot can only sense robots that are in front of it, it will not recognize another robot colliding it from the back.

When a joint collision occurs and a robot cannot recognize it - there is nothing this robot can do but keep doing what he did. On the other hand, whenever it knows it collided it must do something. If it collided in passive time, it should coordinate. The main question is what it should do when it is in active time and it collides once again with a robot. One option is to keep coordinating as if no collision occurred. Another option is to preempt the current coordination method and choose either the same or a new coordination method. According to our modeling the rightest way to treat collisions during active time is to preempt the current coordination method and choose a new coordination method.

4.3 Varying active and passive times

In Section 3 we have assumed that outcomes of a collision rely only on the joint action selected. In terms of a WLU approximation it includes all of its elements - A_i, P_i and A_i^ϕ. It can be directly inferred that the cycle length $l = A_i + P_i$ will stay the same given a joint collision. However, a robot does not know the joint action played and from its viewpoint the active and passive times it will obtain will vary, even if it chooses the same individual action repeatedly because the active and passive times also depend on other robots. In addition, in practice the cycle length may vary even for the same joint action. Therefore, we would like to do some averaging on A_i, P_i and A_i^ϕ and then calculate a WLU approximation which is an averaging over the last collisions $\widehat{WLU} = \frac{\overline{A_i}+\overline{A_i^\phi}}{\overline{A_i}+\overline{P_i}}$.

This can cause inaccuracies in learning WLU approximations because the cycle length $A_i + P_i$ is a real-valued signal in continuous time while sampling of A_i, P_i and A_i^ϕ is discrete. *Semi Markov Decision Processes (SMDP)* [9] are models that represent discrete sampling of a continuous-time reward and also introduce a Q-Learning variant for SMDPs. We suggest a heuristic based on SMDP which

Figure 2: Krembot swarm robots.

is a modification of Q-learning that will overcome this inaccuracy without changing the informational demands of the robots. We call this the *Continuous Time Q-Learning*. The formula is similar to Q-Learning but with some modifications: The learning rate α is set to be a function of cycle length $\alpha \leftarrow 1 - e^{-\frac{\Delta t}{\tau}}$; the bigger the cycle length the closer it will be to 1, thus giving more weight to collisions with longer cycle length. A_i, P_i and A_i^{ϕ} are now also scaled according to cycle length: $A_i' \leftarrow (1 - e^{-\frac{A_i}{\tau}})$, $P_i' \leftarrow e^{-\frac{A_i}{\tau}}(1 - e^{-\frac{P_i}{\tau}}) \cdot P_i$, and $A_i^{\phi'} \leftarrow (1 - e^{(-\frac{A_i^{\phi'}}{\tau})}) \cdot A_o$. Finally, we calculate the Q function $q(x_i, s_i) \leftarrow (1-\alpha)q(x_i, s_i) + \alpha(-\frac{A_i' + A_i^{\phi'}}{A_i' + P_i'} + \gamma \cdot max_{s'}(Q(x_i', s')))$.

4.4 Parallel collisions in different clusters

The actions selected by the robots are not the only factor in the outcome of the system. The model assumes a single synchronous collision, while in reality, robots are spread all over the work area, possibly engaged in multiple parallel collisions. We believe this challenge can be addressed by looking at the density of each collision, but leave this for future work.

5 EXPERIMENTS

5.1 Experimentation environments

Before describing the experiments we conducted, we will introduce the environments we conducted experiments on. We initially started with the *Alphabet Soup* simulator and then moved to experimenting with real robots - the Krembot swarm robots.

5.1.1 Krembot swarm robots. After testing reactive arbitration in the Alphabet Soup simulator we moved to testing the adaptation in a real world environment. We used Robotican's Krembot swarm robots in a domain which is a variant of multi-robot foraging.

Krembot robots are swarm robots with relatively limited sensing and processing capabilities. They are cylindrical-shaped robots with a height of 10.5 cm and a diameter of 6.5 cm. Despite their limited sensing capabilities, those robots can detect collisions and also distinguish between a robot and a static object (see https://www.robotican.net/kremebot).

We tested is a variant of multi-robot foraging. In this domain there are a few stations fixed in position where robots can acquire pucks from. Once a robot reaches one of those stations, it takes the puck and retrieves it to a small area which we call the homebase. It should be noted that since the Krembot robots have no localization capabilities they are unable to either remember or plan a path to one of the stations. Therefore, they do it by randomly searching for a station.

Experimentation settings. In a similar manner to Alphabet Soup, the main objective of the robots is to gather as many pucks to the homebase in a given time. We would like to experiment with reactive arbitration that will maximize this objective.

We experimented with two Best evade coordination methods, one with a time parameter of 500ms and the other with a time parameter of 10000ms. The duration of each run is 1 hour long. For each hour-long we logged each event such as a collision or a puck that was retrieved. From this log we extracted statistics on the number of pucks retrieved and the coordination method choices of the robots. We extracted statistics based only on the last 15 minutes of the run since we want the learning to stabilize.

Using the above configuration, we tested the performance of 4 robots and 8 robots for several configurations. We measure the group performance of each configuration and the time fraction the robots spent on Best evade 500. This time fraction includes the active time and the resulting passive time of choosing Best evade 500. Since there are only two methods, it is easy to derive the time fraction of Best evade 10000.

5.1.2 Alphabet Soup. The Alphabet Soup simulator simulates the multi-robot task of order picking. Order picking is the task of collecting items, usually in a structure like a logistic warehouse, in order to compose orders made by customers. In the Alphabet Soup simulator the items are portrayed as letters and the orders are portrayed as words. It is comprised of several *word stations* where each word station has a list of words to be composed, *buckets* which contain letters, *letter stations* which are used to re-fill buckets with letters and the robots which do all the work. The robots have three main tasks: The first is to take a bucket to a word station in order to put one letter in this station. The second is to return a bucket to its original position and the third is to take a bucket to a letter station.

In this simulator, the task allocation for the robots is centralized and the collision avoidance mechanism is a reactive heuristic which is a combination of dynamic window (moving towards most vacant direction) and waiting in place for a random amount of time. Robots apply this collision avoidance mechanism when they sense that they are too close to other robots.

The task allocation mechanism remains unmodified by us. So is the mechanism which detects and decides when a robot should coordinate. We only modify the coordination mechanism itself. We do it by replacing the original collision avoidance mechanism with a mechanism that arbitrates between reactive coordination methods. It should be noted that since the original collision avoidance mechanism can be treated as a reactive method, it can be included inside the mechanism which arbitrates between reactive methods.

Experimentation settings. The main measurement of performance for this simulator is the amount of letters placed in word stations in a given amount of time. Unless stated otherwise, Each simulation

is 10 minutes long with the last 30 seconds used for measuring performance and other statistics. In addition, unless stated otherwise, measuring n_a is done by collisions and not by density.

5.2 WLU-based Learning

We first evaluated the use of the learning procedure in the Kremotbots. We tested Best Evade 500 and Best Evade 10000 with EI [18] using the same Q-Learning parameters for learning and adaptation. Figure 3 indeed shows that not only learning EI with regular Q-Learning improves performance, but also with more learning variants: EI and Minimum Over Actions approximations, both using continuous-time Q-Learning with $\tau = 10^{10}$ nanoseconds and an exploration rate of 0.02 both improve performance significantly over previous work.

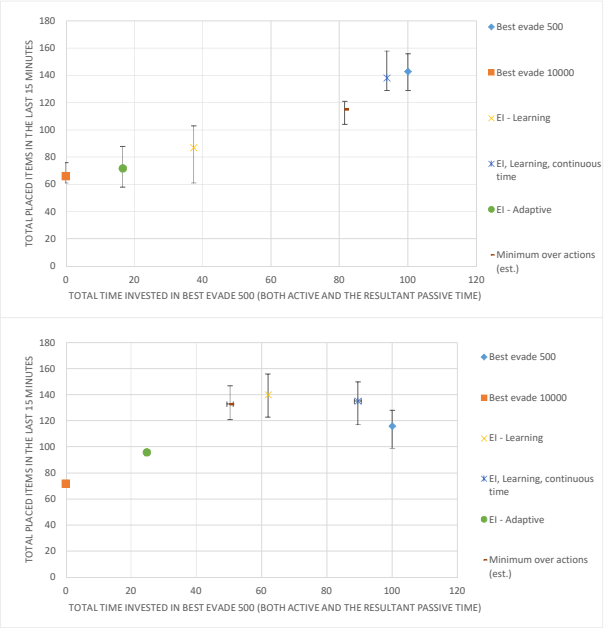

Figure 3: Learning in Krembots for 4 robots (upper chart) and 8 robots (bottom chart).

Even if the learning procedure is not always the best, it can improve performance in comparison to other algorithms. We test two heterogeneous action sets in Alphabet Soup: **(i)** Repel (700), Noise(540), Aggression(500), Original, Best evade(600), and **(ii)** Repel (200), Noise(500), Aggression(2000), Original, Best evade(200). For each of the action sets we compare the Original coordination method to random choice and then to the Minimum Over Actions WLU approximation. For the WLU approximation we used the continuous time Q-Learning algorithm with $\tau = 10^{10}$ nanoseconds and an exploration rate of 0.02. Figure 4 shows that learning using the WLU approximation significantly improves performance in action set (i) and performs slightly better in action set (ii).

5.3 Comparison to global utility rewards

Above, we tested rewards that approximate WLU of EI_{tot}. We know that EI_{tot} is indirectly connected to U. Despite the fact that EI_{tot}

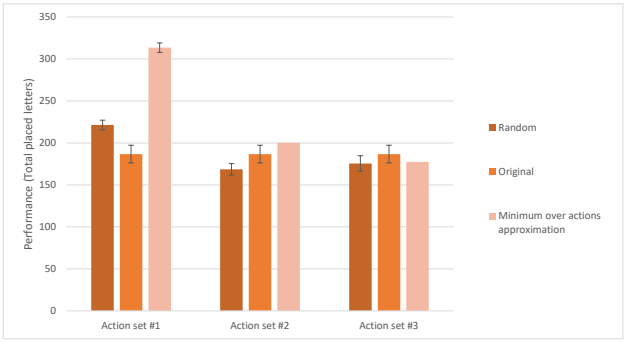

Figure 4: Performance of random choice vs. the original coordination method and the Minimum Over Actions WLU approximation.

is domain independent and U is domain dependent, we would like to test whether this indirect connection may impair performance. Therefore, we compare $WLU(EI_{tot})$ approximations to rewards directly based on U.

In the COIN framework presented in [30], reward functions are based on U. We experiment with those reward functions in Alphabet Soup. In the context of Alphabet Soup, the performance U is the total placed letters. Given a time interval Δt, we denote Δl as the amount of letters the system placed in this time. Therefore $\frac{\Delta l}{\Delta t}$ will simply be the (average) rate of placed letters by the system in Δt. In the same manner we define Δl_i for the amount of letters placed by robot i. Each robot learns rewards based on those variables in the same manner as in reactive arbitration. This implies that Δ_t will represent a robot's interval between collisions. The reward functions used in [30] are:

- Team Game (TG) - The rate of placed letters of all the robots in a given time interval. For robot i, $TG_i = \frac{\Delta l}{\Delta t}$.
- Selfish Utility (SU) - The rate of placed letters of robot i in a given time interval. $SU_i = \frac{\Delta l_i}{\Delta t}$.
- Wonderful Life utility (WLU) - The wonderful life utility of U (not EI_{tot}). We approximate this WLU by the formula $WLU_i = SU_i - \frac{\sum_{j \in col_i} SU_j \cdot A_j}{\Delta t}$ where col_i is the set of robots which collided with robot i in its collision cycle and A_j is the active time of robot j.
- Aristocrat Utility (AU) - This reward is similar to WLU of U. If the main concept of WLU is the difference between the group performance with the robot and without the robot, the AU is the difference between the group performance with the robot performing the current action and the group performance with the robot performing an "average" action. The approximation of the AU is $AU_i = SU_i - \frac{\sum_{s \in S} P_i(s) Q_i(s)}{\Delta t}$ where $P_i(s)$ is the percentage of time robot i chose action s and $Q_i(s)$ is the Q-value of robot i for action s.

We ran simulations with those approximations for 8 minutes and measured the rate of placed letters during the whole run. For each reward function we used Q-Learning with a learning rate of 0.1 and exploration rate of 0.1. The result for TG was averaged over 103 runs, SU over 106 runs, WLU over 109 runs and AU over 112 runs.

Figure 5 shows that the WLU of U outperforms all other rewards based on U. This agrees with the results in [30].

However, given that the WLU of U performs best, we compared it to different approximations of the WLU of EI_{tot}, using different algorithms:

- Minimum Over Actions, using continuous time Q-Learning with $\tau = 10^{10}$ nanoseconds and exploration rate of 0.02.
- EI, using regular Q-Learning, learning rate 0.05, Exploration rate 0.02.
- EI, using WoLF-PHC [8], Exploration rate 0.02, α=0.05, δ_w=0.0005,δ_l=0.005.

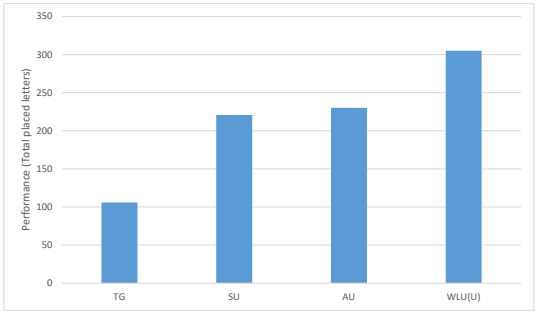

Figure 5: Performance of rewards based on U.

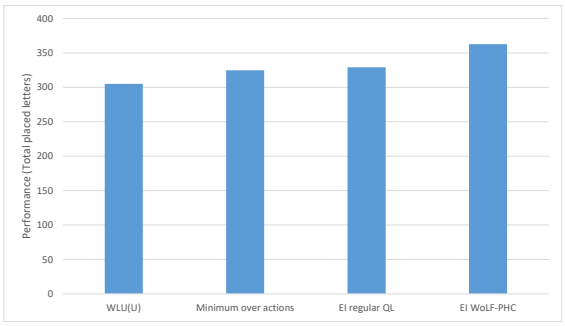

Figure 6: Performance of $WLU(U)$ vs. different approximations of $WLU(EI_{tot})$ with different learning algorithms.

Figure 6 shows that $WLU(U)$ has the lowest performance in comparison to all $WLU(EI_{tot})$ approximations. This shows that consisting directly on U and its derivatives does not improve performance, even though rewards based on EI_{tot} are connected to U only indirectly.

6 CONCLUSIONS

This paper contributed a solid theoretical model of swarm tasks, showing a mathematical connecting between the *Coordination Overhead* (*CO*) of the robots in the group task, to the global utility of the system. We then connected between the *CO* of the whole run to the decisions of robots in a single collision. This allows us to show that in principle, the robots can maximize an individual reward for each collision that will yield optimal global utility in the long run.

We have shown several solutions to challenges that may rise in practice when applying the theoretical model. First, we developed a continuous time variant of Q-Learning in order to address possible inaccuracies of regular Q-Learning that may rise in such domains. Second, we have suggested that the density of a robot can be used as a state space in order for the robot to know how many robots it collides with. Experiments show that the algorithms that we used in order to overcome the gaps in previous work do improve performance while holding guarantees.

REFERENCES

[1] Noa Agmon, Noam Hazon, and Gal A. Kaminka. 2008. The Giving Tree: Constructing Trees for Efficient Offline and Online Multi-Robot Coverage. *Annals of Math and Artificial Intelligence* 52, 2–4 (2008), 143–168.

[2] Noa Agmon, Sarit Kraus, and Gal A. Kaminka. 2008. Multi-Robot Perimeter Patrol in Adversarial Settings. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-08)*.

[3] Gürdal Arslan, Jason R. Marden, and Jeff S. Shamma. 2007. Autonomous vehicle-target assignment: a game theoretical formulation. *ASME Journal of Dynamic Systems, Measurement, and Control* (2007).

[4] Tucker Balch. 1999. The impact of diversity on performance in multi-robot foraging. In *Proceedings of the third annual conference on Autonomous Agents*. ACM, 92–99.

[5] Tucker Balch and Ron C. Arkin. 1998. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation* 14, 6 (December 1998), 926–939.

[6] Maxim A Batalin and Gaurav S Sukhatme. 2002. Spreading out: A local approach to multi-robot coverage. In *Distributed Autonomous Robotic Systems 5*. Springer, 373–382.

[7] Sara Bouraine, Thierry Fraichard, Ouahiba Azouaoui, and Hassen Salhi. 2014. Passively Safe Partial Motion Planning for Mobile Robots with Limited Field-of-Views in Unknown Dynamic Environments. In *Proceedings of IEEE International Conference on Robotics and Automation*.

[8] Michael Bowling and Manuela Veloso. 2001. Rational and convergent learning in stochastic games. In *International Joint Conference on Artificial Intelligence*, Vol. 17. Lawrence Erlbaum Associates, Ltd., 1021–1026.

[9] Steven J. Bradtke and Michael O. Duff. 1994. Reinforcement Learning Methods for Continuous-Time Markov Decision Problems. In *Advances in Neural Information Processing Systems*. MIT Press, 393–400.

[10] Caroline Claus and Craig Boutilier. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI* 1998 (1998), 746–752.

[11] Yinon Douchan and Gal A. Kaminka. 2016. The Effectiveness Index Intrinsic Reward for Coordinating Service Robots. In *13th International Symposium on Distributed Autonomous Robotic Systems (DARS-2016)*, Spring Berman, Melvin Gauci, Emilio Frazzoli, Andreas Kolling, Roderich Gross, Alcherio Martinoli, and Fumitoshi Matsuno (Eds.). Springer.

[12] Yehuda Elmaliach, Noa Agmon, and Gal A. Kaminka. 2010. Multi-Robot Area Patrol under Frequency Constraints. *Annals of Math and Artificial Intelligence* 57, 3–4 (2010), 293–320.

[13] D. Fox, W. Burgard, and S. Thrun. 1997. The dynamic window approach to collision avoidance. *IEEE Robotics Automation Magazine* 4, 1 (Mar 1997), 23–33.

[14] Luca Giuggioli, Idan Arye, Alexandro Heiblum Robles, and Gal A. Kaminka. 2016. From Ants to Birds: A Novel Bio-Inspired Approach to Online Area Coverage. In *13th International Symposium on Distributed Autonomous Robotic Systems (DARS-2016)*, Spring Berman, Melvin Gauci, Emilio Frazzoli, Andreas Kolling, Roderich Gross, Alcherio Martinoli, and Fumitoshi Matsuno (Eds.). Springer.

[15] Julio Godoy, Ioannis Karamouzas, Stephen J. Guy, and Maria Gini. 2015. Adaptive Learning for Multi-Agent Navigation. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multi-Agent Systems*. International Foundation for Autonomous Agents and Multi-Agent Systems, 1577–1585.

[16] D. Goldberg and M. Matarić. 1997. Interference as a Tool for Designing and Evaluating Multi-Robot Controllers. In *AAAI/IAAI*. 637–642. citeseer.nj.nec.com/goldberg97interference.html

[17] Christopher J. Hazard and Peter R. Wurman. 2006. Alphabet soup: A testbed for studying resource allocation in multi-vehicle systems. In *In Proceedings of the 2006 AAAI Workshop on Auction Mechanisms for Robot Coordination*. 23–30.

[18] Gal A. Kaminka, Dan Erusalimchik, and Sarit Kraus. 2010. Adaptive Multi-Robot Coordination: A Game-Theoretic Perspective. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-10)*.

[19] Gal A. Kaminka and Inna Frenkel. 2005. Flexible Teamwork in Behavior-Based Robots. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*.

[20] Jason R. Marden and Adam Wierman. 2009. Overcoming limitations of game-theoretic distributed control. *Proceedings of the 48h IEEE Conference on Decision*

and Control (CDC) (2009), 6466–6471.

[21] D Monderer and LS Shapley. 1996. Potential games. *Games and Economic Behavior* 14, 1 (1996), 124–143.

[22] Ioannis Rekleitis, Ai Peng New, Edward Samuel Rankin, and Howie Choset. 2008. Efficient Boustrophedon Multi-Robot Coverage: an algorithmic approach. *Annals of Mathematics and Artificial Intelligence* 52, 2 (01 Apr 2008), 109–142. https://doi.org/10.1007/s10472-009-9120-2

[23] Avi Rosenfeld, Gal A. Kaminka, Sarit Kraus, and Onn Shehory. 2008. A Study of Mechanisms for Improving Robotic Group Performance. *Artificial Intelligence* 172, 6–7 (2008), 633–655.

[24] Paul Rybski, A. Larson, M. Lindahl, and Maria Gini. 1998. Performance evaluation of multiple robots in a search and retrieval task. In *Proceedings of the Workshop on Artificial Intelligence and Manufacturing*. Albuquerque, NM, 153–160.

[25] F. Sempe and A. Drogoul. 2003. Adaptive patrol for a group of robots. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, Vol. 3. 2865–2869 vol.3. https://doi.org/10.1109/IROS.2003.1249305

[26] Kagan Tumer, Adrian K. Agogino, and David H. Wolpert. 2002. Learning Sequences of Actions in Collectives of Autonomous Agents. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1 (AAMAS '02)*. ACM, New York, NY, USA, 378–385. https://doi.org/10.1145/544741.544832

[27] J. van den Berg, S. Guy, M. Lin, and D. Manocha. 2011. Reciprocal n-body collision avoidance. *Robotics Research* (2011), 3–19.

[28] R. T. Vaughan, K. Støy, G. S. Sukhatme, and M. J. Matarić. 2000. Go ahead, make my day: robot conflict resolution by aggressive competition. In *Proceedings of the 6th International Conference on the Simulation of Adaptive Behavior*.

[29] David Wolpert, Kevin R. Wheeler, and Kagan Tumer. 1999. Collective Intelligence for Control of Distributed Dynamical Systems. *CoRR* cs.LG/9908013 (1999). http://arxiv.org/abs/cs.LG/9908013

[30] David H Wolpert and Kagan Tumer. 1999. An introduction to collective intelligence. *arXiv preprint cs/9908014* (1999).

[31] David H. Wolpert and Kagan Tumer. 1999. *An introduction to collective intelligence.* Technical Report. Handbook of Agent technology. AAAI.

[32] Peter R. Wurman, Raffaello DÁndrea, and Mick Mountz. 2008. Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. *AI Magazine* Spring (2008).

[33] Y. Xu, P. Scerri, B. Yu, S. Okamoto, M. Lewis, and K. Sycara. 2005. An Integrated Token-Based Approach to Scalable Coordination. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-05)*.

[34] Chuanbo Yan and Tao Zhang. 2016. Multi-robot patrol: A distributed algorithm based on expected idleness. *International Journal of Advanced Robotic Systems* 13, 6 (Dec. 2016), 1729881416663666. https://doi.org/10.1177/1729881416663666

[35] Roi Yehoshua, Noa Agmon, and Gal A. Kaminka. 2016. Robotic Adversarial Coverage of Known Environments. *International Journal of Robotics Research* (2016). https://doi.org/10.1177/0278364915625785

[36] Jingjin Yu and Steven M LaValle. 2015. Optimal multi-robot path planning on graphs: Structure and computational complexity. *arXiv preprint arXiv:1507.03289* (2015).