Competitive Ant Coverage: The Value of Pursuit

Alon Shats[1], Michael Amir[2] and Noa Agmon[3]

***Abstract*— This paper studies the problem of *Competitive Ant Coverage*, in which two ant-like robots with very limited capabilities in terms of sensing range, computational power, and knowledge of the world compete in an area coverage task. We examine two variants of the problem that differ in the robot's objective: either being the First to Cover a Cell (FCC), or being the Last to Cover a Cell (LCC). Each robot's goal is to acquire (by visiting first or last, respectively) more cells than the opposing robot, and by that win the game. We examine the problem both theoretically and empirically, and show that the main strategy for dominance revolves around the ability to *pursue*: in LCC, we wish to pursue the opposing robot, whereas in FCC, we wish to create a scenario wherein the opposing robot pursues us. We find that this ability relies more heavily on *knowledge* of the opponent's strategy than on the robot's *sensing* capabilities. Moreover, given the robot's limited capabilities, we find that this knowledge-gap cannot be easily mitigated by learning.**

I. Introduction

The problem of robotic coverage is long-examined in the robotics literature, due to its vast applicability in real-world domains such as agriculture (e.g., harvesting, pest detection), domestic appliances (e.g., floor sweeping, lawn mowing), search and rescue, and security. In this problem, one or more robots are to visit each point in an area at least once to perform some task or detect a change in state. The goal is, therefore, to plan a path for the covering robot(s), referred to as the *coverage path*, while optimizing some criteria, usually minimizing the time to complete the coverage (the *coverage time*). There are numerous variants of this problem, which vary in the knowledge and capabilities of the covering robot(s). We refer to [1] for an excellent survey.

In this work we focus on a recently-introduced form of the coverage problem, *competitive coverage* [2], in which two covering robots operate in an area, but rather than collaborating to cover the area more efficiently, the robots *compete* with one another. In [2] it was assumed that the robots have full knowledge of the environment, thus plan the coverage path in advance. In this work we examine the problem from a new perspective, by assuming that the robots are ant-like robots that have no global information about the environment, and have extremely limited computational, sensing and communication capabilities.

The ant robots, having such limited capabilities and knowledge, cannot plan their coverage path in advance, and must decide on their next move based on what they currently sense in the world, and using a very simple reactive algorithm. We assume the robots can leave pheromone traces in the environment they operate in, and can sense such pheromones in their close surroundings. As in [2], our study focuses on the *asymmetric* version of the *competitive ant coverage* problem, in which only one "competitive" robot (which we control) is aware of the competition and can see the pheromones of the other ant-like robot, whereas the other robot sees only its own pheromones and executes a single-robot optimal coverage algorithm. Asymmetric settings enable us to directly compare strategies which take into account competition and awareness of other robots to strategies that do not, illuminating key differences between competitive and non-competitive settings.

We study two variants of competitive coverage and their relation: First Covered Cell (introduced in [2]) and Last Covered Cell (newly introduced in this work), where the goal of each robot is to be the first or last, respectively, to cover as many cells as possible. This difference in objectives has a significant impact on our analysis.

We first analyze the competitive ant coverage problem theoretically, examining when (and if) a robot can force a win in different environments, robot sensing capabilities and knowledge models. We show that dominating the game revolves around the ability to *pursue*: in LCC, we wish to pursue the opposing robot, whereas in FCC, we wish to create a scenario wherein the opposing robot pursues us. In both FCC and LCC, we identify and prove the correctness of dominating strategies under various starting conditions and in various classes of environments. We further prove that in some settings it is possible to force a win in LCC, whereas the best possible guarantee for FCC is a tie. Following this, we infer that FCC is the more difficult problem for ant-like robots: "pursuing" is easier than "staying ahead".

To complement our theoretical analysis and identify more potential strategies, we model the competitive ant coverage problem as a reinforcement learning (RL) problem attempt to optimize the competitive robot's policy using *Q-Learning* and *Deep Q-Learning*. We show that incorporating the opponent's reward into our reward function during training results in an advantage for developing a winning strategy. We also consider several heuristic algorithms, based on knowing the opponent's strategy (but not their current or starting location), and compare them to each other and to the RL-based strategies. Heuristic strategies prove more consistent, showing that the possession of knowledge of the opponent's strategy cannot be easily mitigated by learning. Finally, both RL-based and heuristic strategies attain significantly higher

*This work was funded in part by ISF grant 1563/22

[1]Alon Shats is with the Department of Computer Science, Bar-Ilan University, Israel alonshats49@gmail.com

[2]Michael Amir is with the Faculty of Computer Science, Technion, Israel ammicha3@cs.technion.ac.il

[3]Noa Agmon is with the Department of Computer Science, Bar-Ilan University, Israel agmon@cs.biu.ac.il

win rates in LCC compared to FCC, further corroborating our theoretical claim that pursuing is easier than staying ahead.

II. Related Work

The problem of Competitive Ant Coverage is drawn from the robotic coverage problem, which has gained considerable attention in the literature [1]. Robotic coverage can be performed in two settings: online and offline. Offline coverage assumes a map of the environment is given, thus the task is mainly to draw an optimal path for the robot such that it passes through the entire area (or multiple paths that jointly visit all points in the case of multi-robot coverage). This approach is meant for robots that are capable of planning and localizing themselves in the area. On the other hand, online coverage does not assume a map exists, but discovers the environment on-the-fly. This approach is more suitable for robots of limited capabilities, and specifically, robots that do not have the ability to keep a map, compute a path, and/or localize themselves in the area. As our focus is on ant-like robots, we work in the online coverage setting.

The problem of cooperative coverage by a group of ant robots was introduced by Wagner et al. [3], offering a graph-based technique for robust coverage. Other graph-based ant coverage algorithms offer better guarantees on maximal coverage time [4], [5], [6], [7]. The works [8], [9], [10] attempt to equalize the size of the area covered by each robot. While all the above works assume the robots have extremely weak capabilities and base their decisions on local sensing (including pheromones), they assume the robots operate in cooperative settings, unlike this work.

Samson and Agmon [2] were the first to introduce *competitive coverage* in an offline setting for two competing robots. In their case, each robot has a map and plans its coverage path in advance, aiming to maximize the number of *first* covered cells. Assuming (similar to our model) an *asymmetric* knowledge setting, they show that only if both the opponent's path and initial location are known in advance, the competing robot has a competitive advantage in terms of *expected* number of First Covered Cells. In this work, we build on these prior results in three ways: (i) by extending competitive coverage to an online setting with ant-like robots, (ii) by investigating environments with *obstacles* ([2] considers only empty grids), and (iii) by introducing the Last Covered Cell model and contrasting it with FCC.

Many works have been written about the use of RL in cooperative coverage, both in offline [11], [12] and online [13], [14] settings. To our knowledge, all existing RL-based multi-robot coverage algorithms, and other learning-based algorithms for ant-like robots, have been designed strictly for cooperative settings, and have not been optimized for competition. In this work we experiment with RL for multi-robot coverage in a competitive setting.

Robotic coverage has been examined in non-cooperative settings in the problem of *adversarial coverage* [15], [16], [17], in which a robot (or a team of robots) should cover an area that contains threats that may stop a robot with some probability. The goal is to compute a path maximizing both the survivability of the robot(s), and the expected covered area. In the adversarial coverage problem the robots have full information of the environment, and strong computational and sensing capabilities. Moreover, the nature of the opponent is inherently different: while in our case both robots are active throughout the game and overtaking the opponent is by covering more cells than it has covered, in the adversarial coverage problem the goal is to cover as many cells as possible before exiting the game.

Gabriely and Rimon [18] describe the online Spanning Tree Coverage (STC) algorithm for a single ant-like robot, resulting in optimal coverage path in grid environments. The mathematics of pursuit between ant-like robots, relevant to our study of LCC, has been investigated in [19].

III. Competitive Ant Coverage: Foundations

In this work we explore the topic of competitive coverage under asymmetric information. In our setting, two ant-like robots, r_A and r_B, compete to cover as much territory as possible, but whereas r_A possesses information about r_B, r_B is completely unaware of the existence of r_A and the fact that it is competing with another robot.

Formally, we assume two robots, r_A (a competitive robot) and r_B (a non-competitive robot), are located at arbitrary *entry points* p_A and p_B, inside a region $\mathbf{R}$ consisting of n discrete locations. $\mathbf{R}$ is assumed to be a connected grid, i.e., a connected subset of size n of the integer grid $\mathbb{Z}^2 = \mathbb{Z} \times \mathbb{Z}$, whose vertices are points (x,y) where x and y are both integers, and (x_1,y_1) is connected to (x_2,y_2) if and only if the Manhattan distance $|x_1-x_2|+|y_1-y_2|$ is exactly 1.

Time is synchronous and discretized to steps of $t = 1,2,\ldots$. At every time step, all robots simultaneously perform a Look-Compute-Move operation sequence, in which they sense their local environment and move to a new location based on a computation they perform.

We follow the traditional ant robotic model for r_A and r_B, meaning they have very limited sensing and computational capabilities. In particular, we assume r_A and r_B are *oblivious*, meaning they possess no persistent memory and move only according to what they see in the current time step. The sensing range is assumed to be V: a robot is aware of vertices in $\mathbf{R}$ that are at a Manhattan distance of V or less from its current position. All robots are assumed to be identically oriented and share the same "north." The robots have no prior map of the environment, nor can they store information they have viewed in the environment - each robot moves only according to what it currently sees. r_A and r_B can choose to leave *pheromones* at their current location - with r_B always doing so. Each location can independently contain one of r_A's and one of r_B's pheromones. Both robots are capable of sensing their own pheromones within their sensing range, and we shall study asymmetric information models wherein r_A can also sense r_B's pheromones (described below).

We consider two competitive coverage games: *First Covered Cell* (FCC) and *Last Covered Cell* (LCC). In FCC, the first robot to place a pheromone on a vertex $(x,y) \in R$ *covers* that vertex and adds it to its territory. In LCC, the *last* robot to place a pheromone on a vertex $(x,y) \in R$ *covers* that vertex

and adds it to its territory. If r_A and r_B enter a vertex they are both eligible to cover at the same time, the vertex is considered to have been covered by both of them. Staying put at a given time step counts as re-entering the same vertex. Both LCC and FCC terminate after n time steps, at which point the robot that covered the most cells wins (or both tie).

r_A's goal is to win over r_B. To characterize the advantage r_A has over r_B when using a given algorithm, we consider two notions of dominance:

Definition 1: The starting conditions of the competitive coverage game are said to be **fair** if $p_A = p_B$ (that is, both robots start the game at the same location).

Definition 2: We say that r_A's algorithm is **dominant** in a region **R** if r_A's territory is guaranteed to be larger than r_B's at termination time. We say that r_A's algorithm is **fair-dominant** in **R** if r_A's territory is guaranteed to be larger than r_B's at termination time assuming $p_A = p_B$.

Whereas r_B is a non-competitive robot and cannot sense the presence of r_A nor r_A's pheromones, r_A is a competitive robot with an *asymmetric information advantage* over r_B. The core type of information advantage we investigate in this work is related to the ability to *pursue*. We shall investigate two pheromone-based models of information:

(i) The *basic model*, denoted I_{basic}, assumes that in addition to its own pheromones, r_A can sense r_B's pheromones whenever they are within its sensing range, and can tell which robot has covered a vertex it sees.

(ii) The *pheromone model*, denoted $I_{pheromone}$, assumes everything I_{basic} does, and in addition assumes that r_A knows how old the pheromones inside its sensing range are. Formally, such that for any pair of locations v_i, v_j in r_A's sensing range which contain pheromones, r_A knows if v_i was created before v_j or vice-versa. Intuitively, this assumption reflects the idea that pheromones might decay over time, such that r_A can distinguish between older and more recent pheromones based on their strength. Practically, this assumption enables r_A to pursue r_B even when it does not see it, as long as r_A senses its pheromones, as it can always move to newer pheromones left by r_B.

Each asymmetric information model opens up different competitive strategies that can be carried out by r_A, hence our results shall vary depending on the model. Some strategies, such as "Lead the Way" (Algorithm 1), do not assume or require any asymmetric information model.

Spanning Tree Coverage: Since r_B is unaware of r_A and/or the fact that it is in competition with some other robot, we assume r_B's main goal is to cover the region **R** as efficiently as possible, i.e., in the least number of time steps. As r_B (like r_A) is an ant-like robot, the optimal coverage strategies available to it are relatively limited. In this work, following the setting of [2], we assume r_B accomplishes coverage by using *Spanning Tree Coverage* (STC) [18]. STC is a type of ant-like coverage algorithm that leaves pheromones on vertices it has covered and requires the robot to move based only on these pheromones (hence does not involve memorizing). It is an optimal coverage algorithm: it can be shown that the robot will move along a Hamiltonian cycle.

STC requires r_B to have sensing range 3 ($V = 3$). Furthermore, and crucially (!), STC assumes **R** can be subdivided into disjoint *cells*, which are square-shaped, 2×2 sub-regions that consist of 4 vertices. It implicitly induces a spanning tree on these cells and causes the robot to "surround" this spanning tree (see Fig. 1). Consequently, throughout this work, *we restrict our analysis to regions which can be subdivided into such cells*.

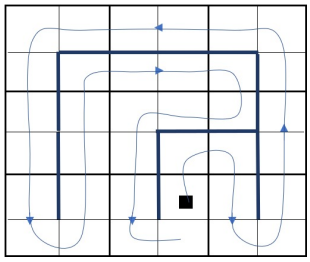


Fig. 1: A spanning tree induced by Spanning Tree Coverage, and the Hamiltonian cycle that it causes the robot to traverse.

Definition 3: The cell containing the vertex $v \in R$ is denoted $cell(v)$.

Definition 4: We shall call the graph whose nodes are cells of **R** and where there is an edge between any two adjacent cells the *cell-level graph*, denoted G_R. We shall call the spanning tree of cells induced by r_B's STC algorithm a *cell-level spanning tree*.

We note that most of our analytical results about possible strategies for r_A do not require that r_B use STC, but remain true in general, assuming r_B moves according to an arbitrary Hamiltonian path of **R**. The exceptions are Propositions 3, 4, 5 and 7 whose proofs rely on properties of STC.

The relation between FCC and LCC: Suppose the path r_B takes from time $t = 0$ until the coverage game ends is $P_1 = v_1 v_2 \dots v_n$, and suppose r_A's path is $P_2 = u_1 u_2 \dots u_n$. Then we have:

Proposition 1: If P_1 and P_2 are Hamiltonian paths of **R**, then r_B wins over r_A in an FCC game if and only if r_A wins over r_B in an LCC game.

Proof: Let q be some vertex in **R**. Since P_1 and P_2 are both Hamiltonian walks, $q = v_i = u_j$ for some i, j. If $i > j$ then q is covered by r_A in FCC and by r_B in LCC, and the reverse is true if $i < j$. Thus the set of vertices covered by r_A in FCC is the set of vertices covered by r_B in LCC, meaning that r_A wins in FCC if and only if r_B wins in LCC. ■

Proposition 1 relates winning scenarios in one coverage game to losing scenarios in the other. This "duality" between LCC and FCC might lead one to think that both games are, in some sense, equally difficult to win from r_A's perspective. However, this is inaccurate for two reasons: first, whereas P_1 is always a Hamiltonian path (due to r_B executing STC or other optimal algorithm), P_2 may contain repeated vertices, and Proposition 1 does not apply in such cases. Second, Proposition 1 talks about predetermined paths–not about algorithms. In fact, both our analytical and empirical results suggest that r_A has an easier time winning in LCC than in FCC.

IV. First Covered Cell

In this section we derive some formal results about r_A's ability to win the First Covered Cell coverage game. We begin with the infinite sensing range setting ($V = \infty$), where we assume r_A can, at any point, see the entire environment **R**. Here we have the following result:

Proposition 2: Suppose $V = \infty$ and the I_{basic} information model. If G_R is not a path graph, r_A has a dominant algorithm.

Note that Proposition 2 being true in the I_{basic} model of course makes it also true in the stronger $I_{pheromone}$ model.

Proof: Let $v_1 v_2 \dots v_n$ be the path robot r_B takes from start to finish of the coverage game, where $v_1 = p_B$. Since r_A has complete visibility of the environment, when r_B is located at v_i, r_A can know its future path $v_{i+1} v_{i+2} \dots v_n$ by simulating STC. Since **R** contains $n/4$ cells, and G_R is not a path graph, G_R's diameter is at most $n/4-2$. It takes r_A at most 2 steps to get from one cell of G_R to any of its adjacent cells, and so r_A needs at most $n/2-4$ steps to get to the cell containing $v_{n/2}$, and at most 2 steps once inside that cell to move to $v_{n/2}$, for a total of $n/2-2$ steps. Meanwhile r_B gets to $v_{n/2}$ after $n/2-1$ steps. Thus r_A can employ the following strategy: get to $v_{n/2}$ before r_B, and subsequently move along the path $v_{n/2} v_{n/2+1} \dots v_n$. This results in r_A covering the last $n/2+1$ of r_B's path before r_B, thus winning FCC. ■

When G_R is a path graph, then, for some initial positions, r_A is able to guarantee at most a tie. Fig. 2 illustrates one such situation: in the first half of the game r_B takes a straight path which is impossible for r_A to get ahead of, thus r_B is guaranteed to cover at least half the locations in **R** first - implying r_A can at most tie with r_B regardless of its algorithm.

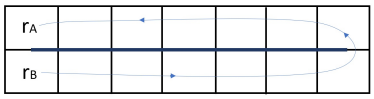


Fig. 2: Following the illustrated path, r_B is guaranteed to tie or win First Covered Cell against r_A.

Let us now consider the case where r_A has only local sensing. STC requires r_B to have sensing range 3. Assuming r_A's sensing range is at least that large ($V \geq 3$), we shall show r_A has a straightforward fair-dominant First Covered Cell algorithm that guarantees a win in any environment with $n \geq 16$ vertices (i.e., four cells).

Let $P = v_1 v_2 \dots v_n$ be the STC path followed by r_B given starting point p_B (so $v_1 = p_B$). Under fair starting conditions, $p_A = p_B$, so r_A can simulate r_B's STC algorithm and always be located at the same location as r_B. In other words, when r_A is located at v_i, it can compute v_{i+1} and move there in the next time step. r_A's fair-dominant strategy, which we call "Lead the Way" (Algorithm 1), relies on the observation that if r_A could skip moving to v_{i+1} and instead move to v_{i+k}, thereon continuing to move according to P, r_A will get a lead on r_B and become the sole captor of the vertices $v_{i+k}, \dots v_n$.

To apply this observation to our current setting, we must ask whether there is ever a situation where r_A is located at v_i and can both compute and move to v_{i+k} for some $k > 1$, in the next step. We do not a priori expect this to be the case, as even computing v_{i+2} may require a larger sensing range than our robots possess, or moving to v_{i+2} may be impossible in a single time step. However, there is a situation where r_A can reliably do this. Suppose for some $i > 0$ that $cell(v_i)$ is a leaf of the (STC-induced) cell-level spanning tree and r_B first enters $cell(v_i)$ by moving to v_i. Upon moving to v_i, a robot executing STC must subsequently cover the other three distinct vertices $v_{i+1}, v_{i+2}, v_{i+3}$ comprising $cell(v_i)$, and has all the sensing data necessary to compute the exact movement sequence through which it will do so. Furthermore, the robot must exit $cell(v_i)$ through v_{i+3}, necessitating that v_i is adjacent to v_{i+3}. This means that instead of moving to v_{i+1} and v_{i+2}, upon entering v_i, r_A can immediately skip to v_{i+3}, getting a lead on r_B in exchange for letting it cover v_{i+1} and v_{i+2}. r_A can then continue executing STC as normal, covering $v_{i+k}, \dots v_n$. This strategy is formally described in Algorithm 1, and we shall prove it guarantees a win for r_A assuming $n \geq 16$.

Algorithm 1 "Lead the Way"

Require: $p_A = p_B$

1: **let** $P = v_1 v_2 \dots v_n$ be the path generated by STC from starting location p_A
2: **let** v_i be r_A's current location.
3: Place a pheromone at v_i
4: **if** $cell(v_i)$ is a leaf of the cell-level spanning tree **and** $cell(v_i)$ has no pheromones of r_A **then**
5: Move to v_{i+3}
6: **else** Move to v_{i+1}
7: **end if**

One can straightforwardly verify that Algorithm 1 is an ant-like algorithm requiring the same sensing range as STC ($V = 3$) and that r_A can always compute v_{i+1}. We note that Algorithm 1 does not require r_A to sense r_B nor its pheromones (i.e., does not require either I_{basic} or $I_{pheromone}$): it merely requires fair starting conditions and the knowledge that r_B is executing STC.

Proposition 3: Assuming $n \geq 16$, Algorithm 1 is fair-dominant.

Proof: (Sketch.) Let $P = v_1 v_2 \dots v_n$ be as in Algorithm 1. Since $n \geq 16$, the cell-level spanning tree contains at least four cells, hence at least one leaf which is not the root of the tree, $cell(v_0)$. Let v_k be the vertex through which r_A first enters such a leaf. After moving to v_k, r_A skips to v_{k+3} and maintains a lead on r_B, but this lets r_B cover 1 more vertex of $cell(v_k)$ than does r_A (v_k is covered by both, v_{k+1} and v_{k+2} by r_B, and v_{k+3} by r_A). Subsequently, r_A will cover all the vertices $v_{k+2}, \dots v_n$ with the exception of vertices of vertices that it skips because they are part of another leaf cell. When r_A enters a leaf cell, it covers two vertices and leaves two vertices untouched which r_B shall later cover, thus neither gaining nor losing territory over r_B. Hence, to show that r_A wins FCC, it suffices to show that there are at least 2 vertices

in $v_{k+2}, \dots v_n$ which do not belong to leaf cells.

Since $n \geq 16$, we know there are at least two cells other than $cell(v_0)$ and $cell(v_k)$ in $\mathbf{R}$. Let us denote these cells c_1 and c_2. We separate the proof into three cases.

If c_1 and c_2 are both leaves, then since $cell(v_k)$ is the first leaf r_A visits (except possibly $cell(v_0)$), we know that r_A not yet entered c_1 and c_2 when it first enters $cell(v_k)$. Furthermore, c_1 and c_2 must have a common ancestor in the spanning tree, c', which r_A must enter at least twice after entering v_k to access c_1 and c_2. During each such visit of c', r_A covers at least one vertex, thus r_A covers at least 2 vertices more than r_B after entering v_k, and wins FCC.

The case where c_1 is a leaf and c_2 is not, and where c_1 and c_2 are both not leaves, are handled similarly. Please see this work's Appendix for the complete details: [20]. ■

When r_A and r_B start in different initial positions, we have the following result:

Proposition 4: Assuming I_{basic} and $V \geq 3$, if G_R is a tree, r_A has a strategy that guarantees it either wins or ties in FCC.

Proof: Let $P_1 = v_1 v_2 \dots v_n$ be r_B's path. If G_R is a tree, then (trivially) $\mathbf{R}$ has a unique cell-level spanning tree. Since r_B is executing Spanning Tree Coverage, P_1 is either a clockwise or counterclockwise Hamiltonian path surrounding the spanning tree (as illustrated in Figure 1). Let us assume it is clockwise, and suppose $p_A = v_k$, for some k. r_A's strategy is as follows: until it is within distance 2 or less of r_B, it executes STC in "reverse", going counterclockwise around the cell-level spanning tree - resulting in the path $P_2 = v_k v_{k-1} \dots v_1 v_n \dots v_{k+1}$. Let us first assume k is even. At time $t = k/2$, r_B and r_A will be at $v_{k/2}$ and $v_{k/2+1}$, respectively. At this point r_A can detect that it is one step ahead of r_B, on r_B's STC path. Thus r_A can begin moving clockwise, covering the vertices $v_{k/2+1} \dots v_n$ before r_B. When r_A and r_B meet, they have both covered $k/2$ vertices. r_A is guaranteed to cover all subsequent vertices upon starting to move clockwise. This guarantees a win for r_A if $k < n$, and a tie when $k = n$. If k is odd, at time $t = (k+1)/2$, r_A and r_B will both be at $v_{(k+1)/2}$ - so r_A can begin moving clockwise, which causes it to always move to the same location as r_B. This guarantees a tie. ■

V. Last Covered Cell

Compared to First Covered Cell, it seems easier to find effective strategies for r_A in Last Covered Cell. This is primarily due to the following difference: the dominant strategies in First Covered Cell rely on covering vertices *before* r_B, which essentially requires r_A to predict r_B's movement path - a difficult task given r_A's limited sensing range. Conversely, the dominant strategies in Last Covered Cell rely on covering vertices *after* r_B, which can be done using pheromones or line of sight.

As in the previous section, we first study possible strategies for r_A assuming $\mathbf{R}$ is completely visible: $V = \infty$.

Proposition 5: Suppose $V = \infty$ and the I_{basic} information model. If $n \geq 8$, then r_A has a dominant algorithm.

Note that the analogous theorem for FCC, Proposition 2, also required that G_R not be a path graph (and Figure 2 demonstrates that this condition is necessary). Proposition 5 does not require this assumption, corroborating our claim that LCC is generally easier than FCC.

Proof: Let $v_1 v_2 \dots v_n$ be the path robot r_B takes from start to finish of the coverage game, where $v_1 = p_B$. If G_R is not a path graph, as shown in Proposition 2, we have that $dist(v_1, v_{n/2}) \leq n/2 - 2$, hence r_A can arrive there before r_B does. r_A can then wait for r_B to arrive at $v_{n/2-1}$, and proceed by executing a pursuit strategy, wherein at every time step t, r_A moves to r_B's current location. This results in r_A covering the vertices $v_{n/2-1} \dots v_{n-1}$ last, for a total score of at least $n/2+1$, thus winning the LCC coverage game.

If G_R is a path graph, we extend the above idea and note that if $n \geq 8$, *one of the vertices* $v_1 v_2 \dots v_{n/2}$ must be at distance less than $n/2-2$ from p_A. Suppose this vertex is v_i. Then r_A can move to v_i in at most $n/2-2$ steps. If $i < n/2$, it can then wait for r_B to get to v_{i+1} if it hasn't already, and subsequently cover $v_i v_{i+1} \dots v_{i+n/2}$ following r_B, winning LCC. If $i = n/2$, it can wait for r_B to get to $v_{n/2-1}$ and subsequently remain one step behind r_B, covering $v_{n/2-1} \dots v_{n-1}$. (The reason this strategy works in LCC but not in FCC is that, if $i < n/2$, r_B might get to v_i before r_A, which, in FCC, would prevent us from covering $v_{i+1} v_{i+2} \dots$). The complete details can be found in this work's Appendix: [20]. ■

We now study the case where r_A has only local sensing. First, let us note that r_A has a simple fair-dominant strategy available to it:

Proposition 6: In the I_{basic} model, assuming $V \geq 1$, r_A has a fair-dominant strategy.

Proof: r_A waits for r_B to leave the entry point and then pursues it. This results in r_A covering $n-1$ vertices. ■

Let us now consider the case where r_A and r_B have limited sensing and have different initial locations. Our only result for First Covered Cell under these conditions was Proposition 4. The situation appears more promising in LCC: not only are we able to strengthen some results compared to those of FCC, we are also able to find new dominant strategies. We start with a stronger version of Proposition 4:

Proposition 7: Assuming the I_{basic} model and $V \geq 3$, if G_R is a tree, r_A has a dominant LCC strategy.

Proof: (Sketch.) Similar to Proposition 4, we have r_A surround the spanning tree counterclockwise until it comes within distance 2 of r_B. Upon meeting, r_A begins pursuing r_B, covering vertices after r_B exits them. If $p_A \neq v_n, v_{n-1}$, this guarantees a win for r_A, by a similar argument as Proposition 4. If $p_A \in \{v_n, v_{n-1}\}$, because $dist(v_n, v_1), dist(v_{n-1}, v_1) \leq 2$, r_A begins pursuing r_B right away, covering all vertices but at most 2, resulting in a win. ■

We now show that, assuming the $I_{pheromone}$ model, r_A has a dominant strategy in environments that can be *fully explored/sensed* in less than $n/2-V$ time steps. We show that this class of environments contains the empty grid and "cycles" as special cases.

Definition 5: The **ball of radius** r about a vertex $v \in \mathbf{R}$, denoted $B(v,r)$, is the set of all vertices at distance r or less from v. The **r-reach** of a path $P = v_1 \dots v_{k+1}$, denoted

reach(P,r), is the set $\bigcup_{v\in P} B(v,r)$.

Definition 6: A path $P = v_1 v_2 v_3 \dots v_k$ is said to ***r*-saturate** the environment $\mathbf{R}$ if *reach*$(P,r) = \mathbf{R}$.

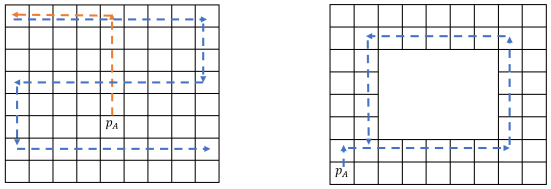
Proposition 8: In the $I_{pheromone}$ model: if, for some $r \leq V$, there exists an algorithm that enables r_A to traverse an r-saturating path of length $n/2 - r$, then r_A has a dominant algorithm.

Proof: Let $P = p_A v_2 \dots v_{n/2-r}$ be an r-saturating path of length $n/2-r$, for some $r \leq V$. Following P, r_A can sense p_B at distance r in time step $n/2-r$, and enter p_B in time step $n/2$. Thereon, r_A can begin following p_B's pheromone path (using its information about pheromone recentness assumed by the $I_{pheromone}$ model). At time $t = n$, r_A will have covered $n/2+1$ vertices after p_B following this strategy (the first $n/2+1$ vertices of p_B's path), winning LCC. ■

Corollary 1: In the $I_{pheromone}$ model, **(i)** Assuming $V \geq 3$, r_A has a dominant strategy in all empty grids of size $m \times m$, for $m \geq 19$ and **(ii)** Assuming $V \geq 2$, r_A has a dominant strategy in environments whose cell-level graph is a cycle.

Proof: **(i)** Figure 3, (a) shows a 3-saturating path of an $m \times m$ grid. The path can be executed by r_A, by first following the orange line to find the top-left corner (this can be done simply by moving up-left, without placing pheromones), and subsequently by zig-zagging down the grid, using pheromone information to keep track of where it is along the path. The orange part of the path is of length at most $2m$, and the blue part is of length $m^2/3+m$, for a total length of $m^2/3+3m$. As long as $m \geq 19$, we have that $m^2/3+3m < m^2/2-3$, and Proposition 8 applies.

(ii) Figure 3, (b) shows a 2-saturating path of length $n/2-2$ in a cycle environment, hence Proposition 8 applies. ■



(a) **(b)**

Fig. 3: (a) A 3-saturating path in an empty grid. (b) A 2-saturating path in a cycle environment.

VI. Heuristic and Learning-based Algorithms

In cases where we could not find a guaranteed winning strategy, we can consider various empirical or heuristic strategies that win most of the time. In this chapter, we examine three different types of algorithms for r_A:[1]

1) *Heuristic.* Family of algorithms that are pre-determined and do not require any prior learning.
2) *Q-Learning.* Algorithms obtained by training r_A using Q-Learning.
3) *Deep Q-Learning (DQN).* Similar to Q-learning, but approximates the Q-value function using a deep neural network rather than a table.

[1]Code for all our algorithms is available at `https://github.com/shatsss/competitive-ants`.

We evaluated the performance of these types of algorithms on three different environments: an empty grid, and two environments with obstacles (see Fig. 4). In each environment, r_B runs *STC* without being aware that it is competing against r_A. Each data point in this section represents the average win rate of r_A across 500 FCC or LCC games. We randomly generate r_A and r_B's initial locations in each game. All (heuristic and learned) algorithms assume the I_{basic} information model. The sensing radius used for r_A is 3 in all Heuristic algorithms, and 1 or 3 in learning-based algorithms.

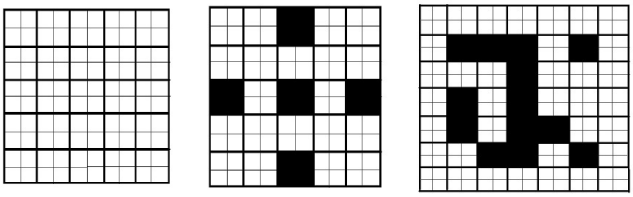


Fig. 4: Different environments: obstacles-free (10×10), obstacles1 (10×10) and obstacles2 (14×14).

A. Heuristic Algorithms

The aim of this section is to compare the effectiveness of various heuristic strategies in FCC and LCC. These algorithms are based on the *STC* algorithm, modified to improve performance in a competitive setting.

1) FCC Algorithms: The first algorithm, FCC_{free}, prioritizes the exploration of cells not yet covered by either r_A or r_B, continuing to visit them until all neighbors have been visited. The second algorithm, $FCC_{R-ahead}$, is motivated by Proposition 4. $FCC_{R-ahead}$ runs STC in reverse (with regards to r_B's STC algorithm) until r_A meets with r_B, after which r_A aims to be one step ahead of r_B or at least to be in the same location as r_B. The third algorithm, FCC_{ahead}, operates the same way as $FCC_{R-ahead}$, but executes STC normally rather than in reverse.

As we can see in Table I, $FCC_{R-ahead}$ significantly outperforms the other algorithms.

TABLE I: Percentage of wins in FCC (rows 1-3) and LCC (rows 4-6) using heuristic algorithms. Each data point is the average of 500 games

		Environment		
		obstacles-free	obstacles1	obstacles2
Algorithm	FCC_{free}	0.823	0.455	0.598
	FCC_{ahead}	0.587	0.52	0.556
	$FCC_{R-ahead}$	0.873	0.906	0.828
	$LCC_{opponent}$	0.994	0.7486	0.953
	LCC_{behind}	0.972	0.806	0.883
	$LCC_{R-behind}$	0.989	0.986	0.891

2) LCC Algorithms: The first algorithm, $LCC_{opponent}$, prioritizes visiting cells that were covered last by r_B. In situations where no neighboring cells have been covered by r_B, the standard *STC* algorithm is applied using r_A's pheromones to resolve the issue.

The second and third algorithms, LCC_{behind} and $LCC_{R-behind}$, are analogous to FCC_{ahead} and $FCC_{R-ahead}$, but instead of staying ahead of r_B, they aim to stay one step behind r_B. $LCC_{R-behind}$, in particular, is described in and motivated by Proposition 7.

In contrast to the FCC setting, where the $FCC_{R-ahead}$ algorithm dominated, the results in Table I show that none of our Heuristic LCC algorithms consistently outperform. $LCC_{opponent}$ outperformed the other algorithms in two out of three environments, whereas $LCC_{R-behind}$ had the highest average win-rate.

Our heuristic algorithms are relatively straightforward, based primarily on pursuing r_B or predicting its next move. This knowledge enables them to attain an excellent win rate, despite their simplicity, highlighting the value of knowing your opponent's strategy in the competitive coverage setting.

B. Reinforcement Learning

In this section, we attempt to learn competitive coverage strategies by leveraging the interaction between r_A and its environment, giving rewards or penalties for certain actions. We test both Q-Learning and Deep Q-Learning as methods of optimizing a competitive coverage policy to be used by r_A. After the training process, r_A follows the resulting policy.

We tested different reward functions during the training process. The first reward function, denoted $T(2)-T(1)$, is based on the scores of both players, where $T(2)$ is the score difference of r_A between the current time step and the previous time step, and $T(1)$ is the score difference of r_B between the current time step and the previous time step. The second reward function, denoted *T(2)*, rewards r_A based on its own scores only, as previously described. The third reward function, denoted *FCC-local*, rewards 1 for moving to unvisited locations, and rewards -0.25 for moving to locations already covered by r_A or r_B. The last reward function, denoted *LCC-local*, rewards 2 for moving to cells already covered by r_B, rewards 1 for moving to unvisited cells, and rewards -0.25 on cells r_A covered last. All reward functions give a score of -100 for illegal moves, such as being an obstacle or outside of the grid, and the game is stopped when such moves occur.

Separate models were trained for each reward function and environment, and the models' respective win rates were calculated and averaged. The initial locations of r_A and r_B were randomized in each training iteration. We note that in actual runs of FCC or LCC, r_A cannot compute the functions $T(2)$ or $T(1)$ (as it has no global knowledge of its territory) - these reward functions are not inputted to the learned policy, just used to optimize it. Table II summarizes the hyper-parameters used for all (deep and normal) RL algorithms.

TABLE II: Hyper-parameters for both Q-Learning and DQN.

Learning rate	0.001
Network update frequency	25 steps
Minimum exploration rate	0.0
Initial exploration rate	1.0
Batch size	128
Experience replay buffer size	50000
Number of training iterations	10000

1) Q-Learning: The input to our Q-Learning model is a single frame that consists of what r_A sees within the sensing range relative to its current location: obstacles, r_B's location if it falls within the sensing range, as well as who visited the cells first/last. We found that this model quickly runs into the problem of *combinatorial explosion*: the number of possible input states grows exponentially with the number of features (determined by the sensing range in our case) used to describe a state, making it difficult or impossible to learn an optimal policy efficiently. For example, when the sensing range is 3, there are 25 neighboring cells in r_A' sensing range, and each cell can be one of the six options (obstacle, not visited, covered by r_A, covered by r_B, covered by both - r_A and r_B, or r_B's current location), for a total of 6^{25} possible states. Consequently, when setting the sensing range to 3, the algorithms' winning rate was close to zero. Therefore, Table III presents the results for simulations conducted with sensing range 1. Hyperparameters are outlined in Table II.

We can see from these results that the $T(2)-T(1)$ reward function outperforms the other reward functions in both games. We find this interesting, because $T(1)$ and $T(2)$ are global functions that cannot be computed by r_A due to its limited sensing range, and are not accessible to r_A when running the trained policy–thus $T(2)-T(1)$ is a noisy reward, sometimes awarding different scores to r_A for doing (as far as it can tell, based on its sensing range) identical actions. Despite this noisiness, it appears beneficial.

TABLE III: Percentage of wins in FCC (rows 1-3) and LCC (rows 4-6) using Q-Learning. Each data point is the average of 500 games

		Environment		
		obstacles-free	obstacles1	obstacles2
Reward	T(2)-T(1), radius 1	0.736	0.47	0.54
	T(2), radius 1	0.600	0.416	0.51
	FCC-local, radius 1	0.662	0.43	0.48
	T(2)-T(1), radius 1	0.996	0.754	0.962
	T(2), radius 1	0.702	0.32	0.44
	LCC-local, radius 1	0.888	0.41	0.75

2) Deep Q-Learning (DQN): When using Q-Learning, we faced a *combinatorial explosion* problem. We therefore attempt to address this problem with Deep Q-Learning (DQN), showing herein its effectiveness in certain scenarios. The input to the model is defined by four frames of the same size that we used for Q-Learning. The first frame contains r_A's location and contains r_B's location if it falls within the sensing range. The second frame contains information about obstacles that r_A detects. The third and fourth frames record the information regarding who covered the cells first and last according to the FCC/LCC game respectively.

Our DQN model, summarized in Fig. 5, includes two convolutional layers with 32 and 64 filters, respectively, each with a stride of 1 and a filter size of 3×3. The first two layers are followed by two fully connected layers, with 128 rectifier units in the first hidden layer and 4 rectifier units in the output layer. The output layer indicates the next move of r_A,

either UP, DOWN, LEFT, or RIGHT. The model uses SELU as the activation function in all layers. Hyperparameters are outlined in Table II.

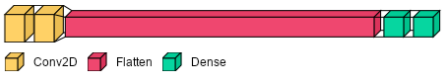


Fig. 5: Overview of the architecture of the DQN model.

An analysis of Table IV reveals that in most scenarios, DQN performs better when we increase the sensing range. "Global" reward functions (i.e., $T(2)-T(1)$ and $T(2)$) are again found to be the most effective for both games in most environments, despite their aforementioned noisiness.

TABLE IV: Percentage of wins in FCC (rows 1-6) and LCC (rows 7-12) using DQN. Each data point is the average of 500 games.

Reward, Sensing range	Environment: obstacles-free	obstacles1	obstacles2
T(2)-T(1), radius 1	0.731	0.25	0.211
T(2), radius 1	0.726	0.33	0.321
FCC-local, radius 1	0.729	0.276	0.24
T(2)-T(1), radius 3	0.948	0.49	0.25
T(2), radius 3	0.93	0.29	0.22
FCC-local, radius 3	0.748	0.42	0.18
T(2)-T(1), radius 1	0.955	0.63	0.84
T(2), radius 1	0.646	0.24	0.15
LCC-local, radius 1	0.965	0.66	0.894
T(2)-T(1), radius 3	0.997	0.911	0.854
T(2), radius 3	0.615	0.3595	0.077
LCC-local, radius 3	0.995	0.876	0.91

The results from our experiments suggest that FCC is more challenging than LCC, as evidenced by consistently lower win rates across all algorithms tested. Finally, the results show that no algorithmic approach consistently outperforms the others, as indicated in Table V.

TABLE V: Best algorithm type per game and environment.

Game	Environment: obstacles-free	obstacles1	obstacles2
FCC	DQN	Heuristic	Heuristic
LCC	DQN	Heuristic	Q-Learning

VII. Conclusions

We studied two types of competitive coverage games for ant-like robots, First Covered Cell and Last Covered Cell, in an asymmetric information setting where one robot has an information advantage over the other. We found that in FCC, we are incentivized to stay *ahead* of our opponent, getting to their next target location before they do, and in LCC we are incentivized to pursue our opponent. Whereas pursuing a robot requires only a line of sight or knowledge of its pheromone traces, staying ahead of a robot - and having it pursue us - requires knowledge of its strategy. Our results suggest pursuing is easier: theoretically, we were able to identify dominant algorithms for LCC in a greater variety of settings than FCC, including settings where we showed no dominant algorithm exists for FCC (Figure 2); empirically, both heuristic and RL-based algorithms had a higher win rate in LCC than in FCC. We are very interested in future results about competitive coverage with symmetric information.

References

[1] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[2] M. N. Samson and N. Agmon, "Competitive coverage:(full) information as a game changer," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6633–6640.

[3] I. A. Wagner, M. Lindenbaum, and A. M. Bruckstein, "Distributed covering by ant-robots using evaporating traces," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 5, pp. 918–933, 1999.

[4] S. Koenig, B. Szymanski, and Y. Liu, "Efficient and inefficient ant coverage methods," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 41–76, 2001.

[5] I. A. Wagner, Y. Altshuler, V. Yanovski, and A. M. Bruckstein, "Cooperative cleaners: A study in ant robotics," *The International Journal of Robotics Research*, vol. 27, no. 1, pp. 127–151, 2008.

[6] O. Rappel, M. Amir, and A. M. Bruckstein, "Stigmergy-based, dual-layer coverage of unknown regions," in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 2023, pp. 1439–1447.

[7] M. Amir and A. M. Bruckstein, "Fast uniform dispersion of a crash-prone swarm." *Robotics: Science and Systems*, 2020.

[8] G. Elazar and A. M. Bruckstein, "Antpap: Patrolling and fair partitioning of graphs by a(ge)nts leaving pheromone traces," *CoRR*, vol. abs/1608.04511, 2016. [Online]. Available: http://arxiv.org/abs/1608.04511

[9] Y. Elor and A. M. Bruckstein, "Multi-a (ge) nt graph patrolling and partitioning," in *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol. 2. IEEE, 2009, pp. 52–57.

[10] B. Ranjbar-Sahraei, G. Weiss, and A. Nakisaee, "A multi-robot coverage approach based on stigmergic communication," in *German Conference on Multiagent System Technologies*. Springer, 2012, pp. 126–138.

[11] L. Piardi, J. Lima, A. I. Pereira, and P. Costa, "Coverage path planning optimization based on q-learning algorithm," in *AIP Conference Proceedings*, vol. 2116, no. 1. AIP Publishing LLC, 2019, p. 220002.

[12] Y. Hu, L. Yang, and Y. Lou, "Path planning with q-learning," in *Journal of Physics: Conference Series*, vol. 1948, no. 1. IOP Publishing, 2021, p. 012038.

[13] O. Saha, G. Ren, J. Heydari, V. Ganapathy, and M. Shah, "Online area covering robot in unknown dynamic environments," in *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*. IEEE, 2021, pp. 38–42.

[14] W. Li, T. Zhao, and S. Dian, "Multirobot coverage path planning based on deep q-network in unknown environment," *Journal of Robotics*, vol. 2022, 2022.

[15] R. Yehoshua, N. Agmon, and G. A. Kaminka, "Robotic adversarial coverage: Introduction and preliminary results," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 6000–6005.

[16] R. Yehoshua and N. Agmon, "Online robotic adversarial coverage," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3830–3835.

[17] R. Yehoshua, N. Agmon, and G. A. Kaminka, "Robotic adversarial coverage of known environments," *The International Journal of Robotics Research*, vol. 35, no. 12, pp. 1419–1444, 2016.

[18] Y. Gabriely and E. Rimon, "Spanning-tree based coverage of continuous areas by a mobile robot," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1, pp. 77–98, 2001.

[19] M. Amir and A. M. Bruckstein, "Probabilistic pursuits on graphs," *Theoretical Computer Science*, 2019.

[20] A. Shats, M. Amir, and N. Agmon, "Competitive ant coverage: The value of pursuit - appendix," 2023. [Online]. Available: https://u.cs.biu.ac.il/~agmon/AlonIROS23-Sup.pdf