Master Thesis

Swarming Bandits: A Rational and Practical Model of Swarm Robotic Tasks

Author:
Eden R. Hartman

Supervisor:
Prof. Gal A. Kaminka

A thesis submitted in fulfillment of the requirements
for the Master's Degree in the Department of Computer Science
Bar Ilan University

May 2022

This work was carried out under the supervision of Prof. Gal A. Kaminka. Department of Computer Science. Bar Ilan University.

Acknowledgments

This thesis summarizes a significant experience for me, I had the privilege to meet and work with wonderful people on topics that I found fascinating, and for that I am grateful.

First, I would like to thank my advisor and friend, Professor Gal Kaminka. Ever since I met Gal, he has always been there for me, willing to help with any problem, from the smallest to the largest, with so much patience and without any judgment. I learned so much from working with him, including how to explain my smart-but-messy thoughts, how to examine my ideas, how to determine what to do next, and so much more. In addition to the many things he contributed to my professional development, he always put my personal development first. He encouraged me to trust my intuition, made me feel that he believed in me and pushed me to believe in myself.

I would also like to thank my team members within the MAVERICK group for creating such an enjoyable and productive atmosphere. Special thanks to Eyal Weiss who kindly and patiently assisted us with some of the math issues, and in general, for his smart and valuable advice. My sincere appreciation to Dr. Hana Weitman, who was always there for me, supportive, helpful and a pleasure to speak with. My gratitude also goes to Chen Rozenshtein, who always shares with me many tools, advice and support, it is greatly appreciated.

Lastly, and most importantly, to my family, especially to my parents, Haim and Mazal, words cannot describe how lucky I feel to have grown up with your endless love and support. Only now, as an adult, I can truly understand and appreciate everything you did and do for me. The way you constantly encouraged me and pushed me to ask questions and to take on challenges, has made me the person and researcher I am today. Thank you for always being there to listen and to help, for always thinking about the best advice, and for always making me feel that I am never alone.

Contents

English Abstract

A *swarm* is a multi-agent system in which robots base their decisions only on *local* interactions with the other robots and the environment. Local interactions limit the robots' abilities, allowing them to perceive and act only with respect to a **subset** of the other robots, and preventing them from coordinating explicitly with all members of the system. Despite these challenging constraints, swarms are often observed in real-world phenomena, and inspired technology for many robotics applications. A key open challenge in swarm research is to be able to provide guarantees on the global behavior of the swarm, given their individual decision rules and local interactions. The reverse is also an open challenge: given the required guaranteed global behavior, how should the individual behave and make decisions?

This thesis proposes a new game-theoretic model for swarms. It ties local decision-making with theoretical guarantees of stability and global rewards. Using simple reinforcement-learning with a reward that is computed locally by each robot, it is able to make guarantees about the emerging global results. Specifically, we show that the utility of the swarm is maximized as robots maximize the time they spent on their task. This allows each single robot to evaluate the efficacy of a collision-avoidance action based on the time it frees up for its own swarm task execution. We use a *multi-arm bandit framework* to allow each individual agent to learn the collision-avoidance actions that are best. Then, we show how to shape the reward used in the learning process, so that it takes into account the marginal contribution of the robot to the swarm. While the marginal contribution is not directly accessible by the robot, it can be approximated effectively based on its own experience. We evaluate the model empirically, using a popular 3D robotics physics-based simulation, in which a cooperative swarm is engaged in *foraging*, a popular canonical task. We compare the results to those achieved by the state of the art, and show superior results.

List of Figures

Chapter 1

Introduction

A *swarm* is a multi-agent system in which robots base their decisions only on *local* interactions with the other robots and the environment [69]. Local interactions limit the robots' abilities, allowing them to perceive and act only with respect to a **subset** of the other robots, and preventing them from coordinating explicitly with all members of the system. As a result, common multi-agent tasks such as task allocation and group planning are particularly challenging: not only are these tasks to be carried out in a distributed fashion (as in all multi-agent systems), they have to be carried out with no global communications or interactions.

Despite these challenging constraints, swarms are often observed in real-world phenomena [29, 5, 4, 23]: in crowds of humans, schools of fish or flocks of birds, insect aggregates (bee hives, ant and termite colonies, locust), and in bacteria aggregates. These inspired technology in use in many applications [47, 18, 26, 69, 60, 8, 34, 21]: computer graphics, simulations, autonomous robots, and medical molecular robotics.

The successful coordination of swarms in all of these instances has been of interest to researchers for many years, in particular attempting to analyze how individual decision-making, combined with strictly local interactions, raise stable global phenomenon of interest. An open challenge in swarm research is to be able to provide guarantees on the global behavior of the swarm, given their individual decision rules and local interactions. The reverse is also an open challenge: given the required guaranteed global behavior, how should the individual behave and make decisions?

This thesis proposes a new game-theoretic model for swarms. It ties local decision-making with theoretical guarantees of stability and global rewards.

Using simple reinforcement-learning with a reward that is computed locally by each robot, it is able to make guarantees about the emerging global results.

Specifically, we show that some swarms can be modeled as a *Stochastic Games* [55]. In this type of game, a sequence of simple (normal-form) games is played, with the result leading, with some probability, to another normal-form game of the same or different type, and so repeatedly. Each stage (normal-form game) in the sequence is an interaction, where robots are in conflict (in our case, they are about to collide). Given the new model, robots in the swarm should be able to rationally select individual actions that lead to equilibrium solutions that are inherently stable, and possibly optimize other criteria (such as social welfare, in the case of cooperative swarms).

However, a game-theoretic model of the swarm is *descriptive*. It is not necessarily useful in guiding the actions of the robots. In particular, when colliding (i.e., when engaging in a stage in the sequence), robots can take one of various actions to resolve the collision. A descriptive model based on the effects of their joint actions on the swarm may indicate better choices. However, the robots might not be able reason and make these choices, because of their inherent limitations.

Thus, several challenges are raised when using the proposed model in practice. These are discussed briefly below, and in detail in the next chapters.

First, the robots do not have global information, and thus are not able to track the swarm's global progress towards its task. In most settings, they also cannot predict the effects of a collision on the swarm's goals, and are therefore unable to compute the joint payoff of the swarm from their individual actions.

Second, in many cases, robots are unable to perceive the actions and payoffs of others. They choose an individual action, but do not know what action the other robots have taken. Thus not only are they blind to the effects of their choices on the swarm's goals, they also are blind to their effects on those with whom they collided, or how the actions of others affect themselves.

Third, the environment itself, even devoid of others, introduces stochasticity in the results of actions. A movement towards some direction may be successful in one location, and less so in another, where an obstacle lies ahead. Thus attempts to learn successful choices will need to address stochastic results from applying them. This is exacerbated when multiple robots are involved, as they introduce additional dynamic changes which, to the individual robot, are perceived as even greater stochasticity (though in fact, they

stem from a different source).

To overcome these challenges, we take several steps in turning the descriptive Markov games model into a prescriptive model, that can be used with reinforcement learning to successfully guide the behavior of robots in swarms, to maximize swarm results despite the limitations listed above.

We first transition from attempting to describe the swarm goals in some task-specific measure of utility, to a description grounded in time spent executing the swarm task. In particular, we address settings in which the utility of the swarm is maximized as more robots are engaged in their swarm tasks, and less in collisions. This introduces a measure of utility which is accessible to each individual swarm robot that is able to measure time.

The individual measurement of task execution time allows each single robot to evaluate the efficacy of a collision-avoidance action based on the time it frees up for its own swarm task execution. Given the stochastic nature of the results, we use a *multi-arm bandit framework* to allow each individual agent to learn the collision-avoidance actions that are best.

However, each individual's measurement of its own time it does not overcome the barrier of global knowledge (how much time the collective spends on its tasks), nor local perception (e.g., how much time another robot is wasting on a current collision). To overcome this challenge, we shape the reward used in the learning process, so that it takes into account the marginal contribution of the robot to the collision and the swarm. While the marginal contribution is not directly accessible by the robot, it can be approximated effectively based on its own experience.

The model is explored in the coming chapters. It departs from previous work in this area by Douchan *et al.* [20], both in its explicit addressing of the stochastic nature of the rewards (previous work addressed it ad-hoc), in the reward approximation analysis, and in allowing for multiple types of stage games to be possible (where previous work allowed a single repeating game).

We also evaluate the model empirically, using a popular 3D robotics physics-based simulation, in which a cooperative swarm is engaged in *foraging*, a popular canonical task. We compare the results to those achieved by the state of the art, and show superior results.

The thesis is organized as follows. Chapter 2 discussed background and related work in detail. Chapter 3 presents the basic stochastic games model of a swarm, and shows mathematically how it can be used a descriptive model of how ideally a swarm works, and how this guarantees stability and maximization of the swarm goals. Chapter 4 then transforms the model to be

prescriptive, addressing how it can be used to guide the individual decision making such that the theoretical guarantees hold in practice. Chapter 5 details the extensive experiments carried out in 3D physics-based robot swarm simulation, to empirically evaluate the model presented. The experiments show not only where it succeeds, but also where it fails when its underlying assumptions are not maintained. Chapter 6 discusses the implications of the work and open questions raised, and concludes. The thesis ends with Appendix A which presents the open-source library implementing the various algorithms used in the experiments.

Chapter 2

Motivation and Related Work in Swarm Robotics

We first briefly introduce swarm robotics, an active area of research (Section 2.1), and focus on *Forgaging*, a popular canonical task for swarms, as a motivating example for the challenges addressed in this thesis. We then examine existing machine learning approaches to addressing these challenges, noting differences with the work presented in this thesis (Section 2.2).

2.1 Swarms and the Challenges They Raise

A group of agents can be referred to as a multi-agent system when individuals in the group interact with each other: they take actions that affect others in the group, they are affected by the actions of others, and they may be able to perceive the existence of others, and the effects of interactions with others. They may be able to communicate information to each other. Implicit in this definition is that the system is distributed: each agent is responsible for its own computation[1].

A swarm is multi-agent system, where the interactions are *strictly limited* to relatively small subsets of the group, i.e., an agent can interact with only a small number of individuals at a time (these are called *neighbors*). So while agents may participate in the swarm to achieve a common goal, they are not

[1]Agents may still utilize centralized computation for making decisions, but this is done by using communications to allow one member to carry the burden of decision-making for the others.

able to communicate or interact globally (i.e., with the entire group at once).

Swarms are of interest to a variety of fields, as they arise both in nature as well as in synthetic systems. For instance, biologists study the swarming behavior of bacteria, insects and complex animals such as fish and birds [48, 11, 38, 4, 6, 5, 21]. Others investigate swarming in humans, as can be seen in crowds, car traffic and pedestrians [29, 23, 59, 45, 64]. Swarms are of interest in simulation and graphics [47, 17, 30, 67, 33, 35, 3], in investigating theoretical distributed systems [2, 56], autonomous vehicles [63, 21, 13, 63], and other applications [27, 34].

Swarm robotics is an area of research within robotics and artificial intelligence, that studies swarms whose agent members are robots. The focus on robotic agents grounds the types of interactions that are considered and their limitations. Commonly, swarm robotics deals with robots that have no or limited communications with their neighbors [69, 60, 18], and may or may not be able to perceive their existence, or their actions and motions.

This thesis focuses on *cooperative* robot swarms (where robots work towards a common goal), and henceforth, when we refer to swarms we specifically mean cooperative swarms. We are specifically interested in robots that are homogeneous in the sense that they have the same capabilities and decision-making algorithms. It is assumed that the environment is unknown.

The common key challenge in swarm tasks is to be able to predict (and guarantee, as the robot designers) the global behavior of the swarm, based on the individual decision rules, which are inherently limited to governing local interactions. The challenge comes up in different variants:

- Given individual decision rules, predicting the global swarm behavior and effects (on some global variables of interest). This is sometimes called the *local to global* problem.

- Given a target global goal (or global optimization criteria), synthesize individual decision rules that guarantee the global outcome. This is analogously called the *global to local* problem.

There are too many investigations of these challenges for a detailed overview, ranging from theoretical studies (e.g., [16, 62, 56]) to highly successful empirical investigations (e.g., [52, 26]). We point the interested reader to recent surveys [60, 8].

Instead, we wish to highlight a popular canonical swarm task, called *Foraging*, which we we use to both illustrate the challenges facing swarm

robotics researchers, as well as (later) a testing-bed for the techniques we introduce in this thesis.

Foraging is a task—inspired by nature—whose goal is to collect the maximum number of items of interest (often referred to as *pucks*) in a work area. Most commonly, in multi-robot foraging, robots have a single base from which they exit, and in which they deposit pucks found in the work area. Robots are unable to self-localize in the work area, and do not have a map. They therefore search for pucks on their way out, and search for the base once they collect a puck (one at a time). Typically, robots colliding (or about to collide) cannot communicate with each other, and so must resolve the collision by initiating avoidance motions which may or may not be effective given the actions of the other robots involved in the collision. In general, robots cannot sense the motions of others (other than their distance in near-collision situations), and cannot perceive whether the other robots are carrying pucks. They also cannot perceive the number of pucks collected so far, or how many pucks are left in the work area.

We use foraging to illustrate the challenges raised to swarm robotics researchers, when designing the individual decision-making procedures for the robots, and when attempting to predict the global results. In the case of foraging, we seek a global guarantee that the swarm will consistently collect the maximal number of pucks in the time allowed. Given that interactions only occur during collisions (or near-collisions), we seek individual decision rules that handle collisions such that th global goals are optimized.

Note the inherent difficulties faced by any single robot attempting to make the decision about how to handle an impending collision. The individual robot does not know how many pucks have been collected by the swarm, or where pucks are located; nor does it know where it is located—it can only distinguish between being in a base or outside of it. It does not know how many robots are in the swarm aside from itself, and may not even be able to tell how many robots are involved in the collision with itself, and it certainly cannot tell whether they are holding pucks or not. These are the conditions under which it needs to decide on an action, such that the swarm is better off.

There have been several investigations that have proposed collision-handling procedures that should improve foraging [22, 28, 44, 61, 68]. However, it has been shown that no one method is good for all cases and group sizes [53]. Deciding on a coordination method for use is not a trivial task.

2.2 Machine Learning in Foraging Swarms

A promising approach to addressing the challenge in selecting interaction actions (collision-handling methods) in foraging was proposed by Rosenfeld et al. [50]. They use learning offline to adjust thresholds which allowed agents to select between coordination methods after being deployed. Though relying on cumbersome offline learning ahead of the task, the method demonstrated that it is possible to improve on any single method, by combining them in the same swarm. However, it provided no guarantees on the results.

A followup investigation by Kaminka et al. [32] proposed using reinforcement learning for online adaptive selection between methods. The focus of the investigation was on a specific reward function (called *Effectiveness Index*), the ratio of the amount of time and resources spent handling the last collision, to the total time and resources spent during the collision and the time until the next one. This measures the overhead spent by the agent handling the collision. This reward was used with a stateless Q-learning algorithm (Q-learning with a single state) with a high learning rate (0.5) to quickly adapt the decisions to the recents conditions in which collisions occur.

The study by Kaminka et al. was the first to use reinforcement learning in foraging, but is certainly not the first to use reinforcement learning in multi-robot or multi-agent systems. Mataric [40] describes several different learning setups, with robots receiving joint rewards, individual rewards, etc. She examined these empirically, without discussing the reward function itself or the task model.

Approaching multi-agent reinforcement learning from the game-theoretic side, Claus and Boutilier [14] investigate a repeated-games model of tasks (a special games of stochastic games), and distinguish two types of learning agents: *joint action learners* which know the actions taken by others, and *independent learners* which are blind to the selection of others. They concluded that generally, independent learners using standard Q-learning do not converge to the Nash equilibrium. This is the case here, as robots are unable to perceive the actions of others. Indeed, while Kaminka et al. [32] conjecture as to the guarantees afforded by the Effectiveness Index reward the proposed, they were unable to prove it, and new research established that in practice, the use of the reward did not always optimize the swarm results [19].

Indeed, the translation of reinforcement learning algorithms and reward functions into guarantees on convergence to Nash or other equilibria is a topic of much ongoing research. When learning as part of a group is considered, the

global reward function the agents seek to maximize is usually a function of the individual reward functions (sum, average, etc.). The individual reward functions, however, depend on a global decision (joint action, majority vote, etc.).

One approach is to change the learning algorithm. For instance, within general reinforcement learning Zhang and Lesser [66] use limited communication to overcome specific limitations of independent learners, i.e., make them a little more like joint learners. Littman [37] suggested an algorithm called *Team Q-learning* and proved it convergence to optimal solution in the case of unique solution. Bowling and Veloso [10] introduced a new principle, *"Win or Learn Fast" (WoLF)*, that handles the problem of parallel learning with a variable learning rate, they proved convergence for some cases and demonstrated greater empirical success. Conitzer and Sandholm [15] used the "WoLF" concept as a basis and presented an algorithm named AWESOME (Adapt When Everybody is Stationary, Otherwise Move to Equilibrium), it assumes each agent can observe other agents' actions to determine whether they are stationary, and then learn based on those observations. There are numerous such investigations; Hernandez et al. [31] provide a recent comprehensive survey.

However, these generally require the learning agent to know the actions of the others, or their outcomes. The learning agents are not completely independent. Similarly, most work within *multi-agent multi-armed bandit* learning models assume some level of communication between the agents [54, 12, 39, 1]. For instance, Shahrampour et al. [54] suggested a way for the agents to estimate the global reward with local communication. Chakraborty et al. [12] used global communication for the same purpose, but added a cost for communication to reduce the use of it.

Independent learning agents are particularly challenging, because they learn in isolation and so unwittingly respond to changes in the rewards, that stem for the parallel processes of learning taking place in other agents. Matignon et al. [41] carefully analyzed and distinguished specific challenges to independent learning agents that are raised under specific conditions. One of these challenges is that from the point of view of the individual independent learning agent, the reward distribution is non-stationary: it varies and changes as other agents learn in parallel, but their existence and their decisions are not known to the learning agent. Some algorithms have attempted to focus on this inherent non-stationarity of the reward distribution. For example, variants of the classic UCB1 algorithm (which is optimal for multi-

arm bandits) have been proposed by Garivier and Moulines [24, 25]. We will later empirically compare the results we achieve with theirs.

A different approach to addressing the challenges in multi-agent learning focuses on changing the reward function itself, as Kaminka et al. attempted [32]. Wolpert and Kagan [65] discuss the use of the marginal contribution of each agent on the global utility, as the individual reward to the agent. The call the reward function the *Wonderful Life Utility* (WLU) as it measures the contribution of the agent to the group by contrasting the group utility with and without its action.

Later, Douchan, Wolf, and Kaminka [20] have used WLU together with the Effectiveness Index, to model swarms as repeated games, in which the Nash equilibrium also maximizes social welfare (the aggregate of all agents payoffs). This is the most closely related work to ours. It derives the individual reward from the model, and uses common approximation techniques to account for missing information. In comparison, the model we introduce here is more general (allows multiple types of stage games), and is based on measurement of total accumulated direct work, rather than the overhead of a single stage-game. This results in a much more compact and elegant derivation of the individual reward function, and allows compensating for stochastic settings. Indeed, we demonstrate empirically it improves significantly over the previous approximations and learning model.

2.3 A New Model of Swarms and Its Use with Multi-Agent Reinforcement Learning

Our goal is to find a model that (1) does not rely on any communications, (2) provides *guarantees* on the global behavior and results of the swarm task, and (3) makes realistic assumptions, enough to be *implemented* in practice by simple autonomous robots so that empirical success can be demonstrated.

This thesis addresses the challenges described below, using a combination of a multi-arm bandit algorithm (modified to allow for actions with duration), and a reward derived mathematically from the stochastic games model introduced in the next chapter.

Immeasurable, Task-Dependent Global Utility The goal of the swarm, whether known to the individual swarm member, cannot be mon-

itored or measured by the individual. In foraging, the goal is to maximize the number of pucks collected by the swarm. But the individual agent cannot measure the rate at which pucks are collected, or whether that rate can be improved. Moreover, utility is inherently task- and domain- dependent. Changing the task from foraging to a different task, will require rethinking the utility function, and thus re-shaping the reward function that would be derived from it.

Previous work has repeatedly suggested that minimizing the time spent on the overhead of managing interactions can lead to improved performance [28, 51, 32]. Indeed, under some conditions and assumptions, the total overhead can act as a potential function for the game [20].

We switch the focus from minimizing overhead, to maximizing the direct work (total net working time) spent by the agent, as as a substitute for the specific domain-dependent utility. In game-theory terms, we use time spent on swarm work as the basis for a potential function for the swarm utility. Later, this allows reformulating the individual reward in a manner that is more natural for learning algorithms, and indeed shows improved results.

Unlike the task-dependent utility, time is not only measureable by the robots, but it is inherently task-independent. Any swarm task whose utility can be described as a function of time (as discussed in the next chapters) is directly addressable by the model we present.

Global Information is Not Available Whether pucks or time spent working, the individual swarm robot cannot monitor the progress of the swarm. It cannot count the pucks collected by others; and it cannot measure how much time they have spent working. Because of this, its own assessment of how its choices affect the swarm utility (however it is defined) is inherently and extremely limited.

We take the approach of Douchan et al. [20] by utilizing the marginal contribution of the robot as the basis of an individual reward function which will establish the impact of the action on the swarm. The use of this reward, called the *Wonderful Life Utility* (WLU) was introduced in [65].

This WLU reward determines the marginal contribution of an agent by comparing the utility with and without it. However, the individual agent does not have knowledge of the utilities of others. Thus missing information needs to be approximated. The derivation of the WLU in the framework in previous work was complex, and was carried out using ad-hoc (though fa-

miliar) simple approximation functions (e.g., using the mean, minmax, etc.). Instead, because we compute the WLU of the direct work rather than overhead, derivation of the individual reward was easier to approximate, and indeed more successful as we show empirically.

Stochastic Rewards due to Environment Given that the basis for rewards is the measurement of work time (direct work) and collision-handling time (overhead) after each collision, a persistent challenge is that the rewards can be stochastic in nature. Even when only a single or handful robots are present, the time spent working vs colliding can vary greatly from one collision to the next, simply because of the location of the robots and the shape of the work area: Two robots meeting in corridor with collide with greater frequency than in an open space.

From the perspective of the single robot, the reward on using the same action can vary greatly, making the process slow to converge, at a minimum. Previous work has not addressed this explicitly.

To address the stochastic nature of the rewards, we build on the *Multi-Arm Bandit* framework [57] which inherently addresses the exact problem of learning which action to take when there are stochastic rewards. We treat each interaction method (collision-handling method) a an arm in the multi-arm bandit. The robot is to select a method (pull an arm) that will yield the best expected accumulated result (accumulated work time).

Actions Take Time As rewards in our model are a function of time (e.g., time to move away from a collision), the inconsistency of the world also affects the time spent on each action. The standard reinforcement learning algorithms, which operate in discrete steps (each action takes the same amount of time) do not address this issue. As a result, they introduce biases into the learned results, which prohibit or hurt the results in practice.

Previous work [20] has addressed this ad-hoc, by modifying the Q-learning update function, such that longer cycles are given more weight than shorter ones, for the same chosen action. This was done by manipulating both the learning rate and the introduction of a interval duration hyper-parameter. Introduced as *continuous-time Q-learning*, they described the intuition behind it, but did not provide any guarantees for it. Moreover, because they attempted to learn the overhead associated with each conflict, the solution is not straightforward.

Our selection of the learning algorithm within the multi-arm bandit work likewise needs a change, as MAB formulations similarly assume each arm-pull takes the same amount of time. But in our case, a *pull* now also has a duration, so in choosing an arm, the agent should not select the arm that maximizes its average reward based on **how many times** it pulls this arm, but rather its average reward based on **how long** it plays this arm. We therefore modified the UCB1 algorithm [7] to average by time, instead of the number of times the arm was pulled. The change was quite trivial, as the focus on total direct work lent itself mathematically to this change.

Non Stationarity of Reward Distribution due to Parallel Learning A major challenge in multi-agent learning in general is the fact that the agents are learning in parallel [41, 31]. Consequently, the reward distribution changes through time and is non-stationary. UCB1 assumes a fixed (stationary) distribution over the rewards, so its theoretical guarantees do not apply.

The use of the WLU reward does not explicitly address this issue. We therefore attempted to work with algorithms specifically designed to address parallel learning. In particular, we experimented with *discounted USB* (D-UCB) [24], which uses a discount factor over the rewards to adapt distribution changes. Surprisingly, we empirically show that its use lead to no improvement in the learning compared to the WLU reward use. These results will be shown in later chapters.

Multiple States In Douchan, Wolf, and Kaminka's model [20], only one type of conflict was considered, e.g., a single type of collision. Thus the swarm model is that of a repeated game, where each game is the same type of interaction. However, this is a restrictive assumption in practice. A collision from behind is not the same (and should not be resolved in the same manner) as a collision from the front.

We therefore generalize the previous work to model the swarm as engaging in a Markov Game (stochastic game), where multiple types of collisions or interactions are allowed. Like previous work, we are interested in the total sum of all robots direct work, and seek to minimize their overhead spent on collisions. Unlike previous work, however, we allow for multiple types of interactions to be managed, something which could not be achieved with previous work.

Some collisions may be resolved differently, and more easily, than others. Thus a single strategy (for one type of game), would not be optimal for all types. We theoretically consider the case in which all types are **distributed uniformly**, i.e., agents cannot use strategies that focus on getting to easier conflicts. Later, we also discuss the difficulties to the multi-arm bandit algorithms when the distribution is not uniform.

Chapter 3

A Descriptive Stochastic Games Model of Swarm Tasks

This chapter presents a stochastic games model of swarm tasks. The model is descriptive, in that it shows how swarms may be described and analyzed as stochastic games, but this in itself does not proscribe individual agent decisions in practice. However, it does argue that successful swarms may be viewed as reaching a Nash equilibrium which also maximizes social welfare (the sum of all agents utilities).

In Section 3.1, we provide formal definitions and notations, and explain the fully-cooperative model, which is the basis for our work. In Section 3.2 we explain how the shared utility of the fully-cooperative game, in a distributed swarm, is the sum of individual swarm members' rewards. In Section 3.3, we transform this basic model to using time as a proxy for utility, replacing the domain-dependent utility measure with a measurement of time, which is both task-independent, and accessible to robots in swarms. Section 3.4 shows the model makes assumptions that prohibit its direct use in practice to guide individual robot decisions. These will be discussed in the next chapter.

3.1 Fully-Cooperative Swarm Games

We consider swarms of n homogeneous robots (each with the same capabilities and decision-making algorithms), which act only according to local sensing. The swarm carries out a cooperative task: The goal is to maximize the total utility of all swarm members. The robots are assumed to be co-

operative: they do not seek to minimize their work, but rather to maximize their contribution.

3.1.1 Cooperative Swarms

The need for coordination in swarms comes arises in two distinct situations. First, robots may inadvertently interfere with each other, i.e., they are *in conflict* [32]. In robots, this happens for example when their trajectories cross and they (are about to) collide. A second for of coordination may be needed materially for the task, when robots cannot perform a component of the task by independently of each other, i.e., more than a single robot is needed to carry out an atomic component of a task (e.g., lifting a long table from both ends requires two robots, but lifting a chair may be carried out by a single robot).

We focus on swarm tasks which involve only the first type of coordination (Assumption 1). For instance, in foraging (see Chapter 2), each robot collects pucks on its own, and the swarm benefits from a larger number of total collected pucks. Coordination in foraging is needed only to resolve collisions between robots. Had the definition of foraging require pucks to be carried in pairs to the base, this would have been an example of the second type of coordination.

Assumption 1. *The swarm task requires coordination only to prevent or reduce conflicts between individuals.*

We follow up on previous work [32, 20] in looking at each swarm member's activities as a sequence of transitioning between two states, back and forth: a *Task* state (also called *program* mode, and denoted P) where the swarm members are carrying out the actual task of the swarm, and a *coordination* state (also called *avoidance* state, denoted A), where members are in conflict, and must spend resources avoiding collisions. For each swarm member, each conflict starts a cycle of coordination and task that ends at the next conflict, and the member's activity over time is a sequence of such conflicts (Figure 3.1).

We assume for now that all robots participate in every collision (Assumption 2). This allows us to model the swarm as a process in which all robots participating synchronously. As the conflicts are mutual, their total duration is also mutual. However, the division of the duration into the different states

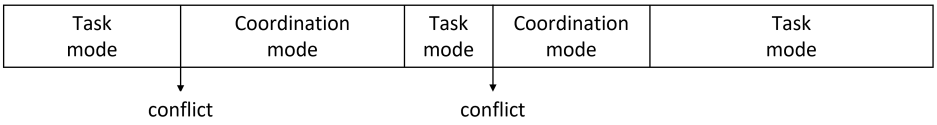

Figure 3.1: A visual illustration of a swarm member's activity timeline. Duration of the *Task* and *Coordination* states are implicitly shown by the length of the respective boxes along the horizontal axis.

of task execution and coordination is not necessarily identical (Figure 3.2). Later, we will relax this assumption, as it does not hold in practice.

Assumption 2. *All conflicts are mutual to all agents.*

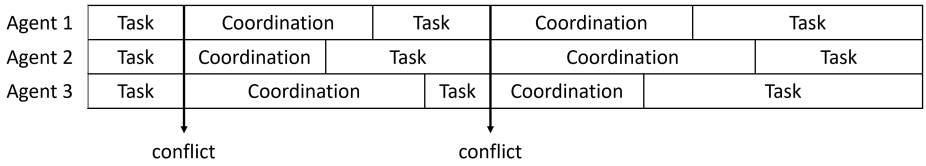

Figure 3.2: A visual illustration of the different divisions of the mutual cycle duration between states according to the different agents.

The swarm cycles from one conflict to the next. With every conflict, the robots choose actions to resolve the collision, so they can go back to carrying out the task. The actions are *joint*: All swarm members enter a collision together, and all exit it together by selecting a joint action. The conflict duration as other outcomes of the conflict resolution are determined by the **joint** action that has been taken.

As robots act individually, the notion of a joint action is that of a combination of the individual collision-avoidance actions of all agents. Each individual robot selects its own response to the collision, and the synchronous combination of all these selections synthesizes the joint action that takes the swarm out of the conflict. Indeed, we assume that every possible joint action can succeed (Assumption 3). This easily holds in practice, as an unsuccessful joint action can be modeled as a joint action that leads to a zero-duration task state, which then immediately transitions into a new conflict.

Assumption 3. *Every joint action can resolve a conflict.*

3.1.2 A Fully Cooperative Game

In this section, we suggest a first model for swarm tasks. Under the assumptions above, we view the swarm as engaging in a *stochastic game* (also known as a Markov game) [55]. Each specific conflict cycle (collision resolution and subsequent task activity) is a normal-form subgame, where robots are *players*, the coordination methods are the possible *individual actions* and the *payoffs* are given by the utility gained during the task state associated with the conflict, and lost during the coordination state. Figure 3.3 illustrates this view.

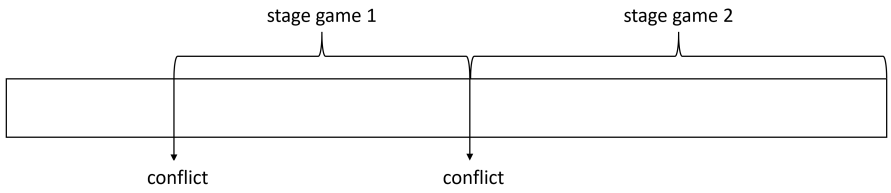

Figure 3.3: An illustration of a swarm task as a sequence of stage games

The swarm task is then viewed as a series of stages. Formally, it is a stochastic game defined by a tuple $\langle N, S, A, D, R \rangle$, where:

- N A finite set of $n := |N|$ players, each a swarm robot.
- S The set of states, each representing a type of conflict that could take place during a stage. In other words, this is the set of stage game types. We use the notation s^k when discussing the specific state of stage k.
- A The set of *joint* collision-avoidance actions (combinations of players' individual actions). See below for more details.
- D $D\colon S \times A \times S \to [0,1]$ is the transition probability function; $D(s,a,s')$ is the probability of transition from state s to state s' after performing the joint action a.
- R $R : S \times A \to \mathbb{R}$ is a function describes the swarm payoff from performing the joint action a at state s.

The joint actions set A. The set of joint actions A is made from the combination of all individual actions A_i, i.e., it is generally defined as $A = A_1 \times A_2 \times \cdots \times A_n$. However, this notation hides salient details.

First, in principle, the set of actions available to each robot may vary between states, so formally we should have defined $A_i : S \to M_i$, where M_i is the set of all actions available to a player, and we should have written $A_i(s)$. For notational brevity, we ignore the state when the exposition does not require it, and simply write A_i.

Second, as the robots are assumed to be homogeneous in their capabilities, $\forall i, M_i = M$, where M is the (same) set of individual collision-avoidance methods available to each player.

The payoff function R. Generally, in stochastic games, there are n **individual** reward functions $(R_i)_{\forall i \in n}$, each function R_i returns the player's individual payoff, having been in state s, where the joint action a was taken (the player's individual action in the state is a component in a).

The definition above uses a more abstract version, where the players' payoffs are joint; all swarm members share the payoff R. This type of game is called a *Fully Cooperative Markov Game* in [14, 41].

The fully cooperative theoretical payoff does not hold in the reality of *independent robots* in swarms. The locality and independence restrictions faced by robots raise the key challenges that are addressed in this thesis: *immeasurability* of the payoff (the units on which the payoff is given are irrelevant to the robots), and the *globality* of the payoff (the robots cannot perceive the payoff of the joint action; at best they can only perceive some local proxy of it). Thus later in this thesis we will transform the model into a more realistic one, where the these challenges will be addressed explicitly.

3.1.3 Solving a Fully Cooperative Swarm Game

Each stage ($t \geq 1$) of a game can be represented as a pair consisting of the state at this stage ($s^t \in S$) and the joint action played in response ($a^t \in A$). A sequence of such pairs is called a *possible play* if there exists a pair for every $t \geq 1$, and the probability of transitioning from any state s^t to state s^{t+1} in the sequence is positive (Definition 3.1.1). The set of all possible plays of G is annotated G_p.

Definition 3.1.1. A *possible play* g of a game G is a sequence of pairs $\langle (s^1, a^1), (s^2, a^2), \ldots, (s^t, a^t), \ldots, (s^T, a^T) \rangle$, where $\forall T \geq t \geq 1$: $D(s^t, a^t, s^{t+1}) > 0$. Note T may be infinite.

Let σ be an assignment of actions **for each player**, for all stages. This set of assignments determines the action each player should take at any stage $t \leq T$. σ is called a *strategy profile* [43]. A strategy profile potentially takes the state-action history up to stage t into account in determining the assignment of the joint-action a^t.

A special case of a strategy profile is one that specifies actions that depend only on the current state s^t, and ignores the history $\langle (s^1, a^1), (s^2, a^2), \ldots, (s^{t-1}, a^{t-1}) \rangle$. Such a strategy profile is called *stationary* [58]. This is the case where $\forall s \in S, t, \exists a \in A$ such that if $s = s^t, (s^t, a^t) \in g$, then $a^t = a$, i.e., the action is selected based only on the state in the stage, regardless of previous states and actions in previous stages. We then denote strategy profile simply as function of the current state, i.e., $\forall t, \sigma(s_t) = a_t$.

A solution in a fully cooperative swarm game. A solution to the fully cooperative swarm game is a strategy profile σ, with which one can induce from a possible play g by applying σ repeatedly. Starting with a given state s^1, applying σ, one generates the joint state-action pair $(s^1, \sigma(s^1))$, which leads to s^2 and so on. For the remainder of this thesis we will restrict ourselves to stationary profiles alone (see Chapter 6 for a discussion).

Where T is the number of stages in the possible play, the accumulating joint payoff of the swarm, when using σ is then given by[1]

$$U^T(\sigma) = \sum_{t=1}^{T} R(s^t, a^t) \cdot D(s^{t-1}, a^{t-1}, s^t) \tag{3.1}$$

$$= \sum_{t=1}^{T} R(s^t, \sigma(s^t)) \cdot D(s^{t-1}, a^{t-1}, s^t) \tag{3.2}$$

For now, we will make the assumption that each state is independent of others, i.e., the probability of transitioning from one state $s \in S$ to a different state $s' \in S$ is uniformly distributed for all joint actions $a \in A$. For swarm tasks, this means that the resolution of one collision can lead to any other collision later on, with equal probability (Assumption 4).

Assumption 4. $\forall s, s' \in S, \forall a \in A$: $D(s, a, s') = \frac{1}{|S|}$.

[1]For $t = 1$, we define $D(s^{t-1}, a^{t-1}, s^t) := \frac{1}{|S|}$, i.e., the start state s^1 is arbitrarily set with uniform probability.

As a result, there is a constant factor $\frac{1}{|S|}$ which is always present when we discuss the rewards and their summation. We will omit it from the formalities for brevity. Chapter 6 will revisit this assumption and notation.

The swarm does not know T. This is equivalent, from a game theory perspective, to an *infinite horizon* game, where T tends towards infinity. Naively then, U grows to infinity and so the import of selecting a is essentially meaningless as long as the reward (described by R) is positive. We therefore follow a standard alternative objective, where the swarm seeks to maximize the *limit of means* of U^T (Definition 3.1.2):

Definition 3.1.2. The utility of a fully cooperative swarm game is given by

$$\begin{aligned} U(\sigma) &:= \lim_{T\to\infty} \frac{1}{T} U^T(\sigma) \\ &:= \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} R(s^t, \sigma(s^t)) \end{aligned}$$

An optimal solution is a strategy profile σ^* that maximizes this utility (Definition 3.1.3). We use U^* to denote the expected utility of σ^*.

Definition 3.1.3. An optimal swarm strategy profile σ^* is one that generates the maximal fully cooperative utility limit of means:

$$\sigma^* := \arg\max_{\sigma} U(\sigma)$$

A second important quality is *stability*, i.e., that once players have a strategy profile, they are incentivized to maintain their individual roles. In game theory terms, a solution that has this quality is said to be in Nash Equilibrium (NE) [42]. The main quality of NE is that agents do not benefit from deviations from the solution's profile, which ensures that the system remains stable.

Theorem 1 shows that a strategy profile that maximizes the long-term swarm payoff reward is also in NE of a full cooperative Markov game.

Theorem 1. *In fully cooperative Markov games, the strategy profile that maximizes the swarm payoff is also a NE of the game.*

Proof. Let $\sigma = (\sigma_1, \sigma_2, \ldots \sigma_n)$ be a strategy profile which maximizes swarm payoff U, where σ_x is the strategy of agent x. Let us denote this maximum utility by $\alpha := \max(U)$. Assume for contradiction that σ is not a Nash equilibrium (NE). This means that there is a player i that benefits from deviating. i.e., it has an alternative strategy as σ_i' which has a higher payoff $\beta > \alpha$. Since the payoff β is identical to all agents in swarm games as defined above, then this means the strategy profile $(\sigma_1, \sigma_2, \ldots \sigma_{i-1}, \sigma_i', \sigma_{i+1}, \ldots \sigma_n)$ yields higher long-term swarm payoff than σ. This contradicts the assumption that σ is the strategy profile that maximizes the payoff. □

Together, stability and joint payoff optimality are excellent qualities for a robot swarm. Stability guarantees on a swarm means that robots will continue their role in the swarm and will not misbehave. And joint reward optimality means the robot swarm will achieve the optimal result. A maximal payoff reward will be stable (see Thm. 1).

3.2 Additive-Utility Swarm Games

The swarm game model introduced above assumes the existence of a function $R\colon S \times A \to \mathbb{R}$ that describes the **joint payoff** at some stage (described by the state at this stage and the joint action taken). It also assumes all agents are aware of the joint payoff. Let us maintain this latter assumption for now, but discuss the individual contributions of the swarm members to the joint payoff.

We modify the definition of the swarm game. We replace the joint payoff function $R : S \times A \to \mathbb{R}$, with a set of individual payoff functions $\mathcal{R} = \{R_1, \ldots, R_n\}$ where $R_i : S \times A \to \mathbb{R}$ is the individual payoff of player i, resulting from the selection of the joint action a in state s.

We then define the stage game payoff R to be the *social welfare utility*, the sum of *individual player payoffs*. $R(s, a) = \sum_{i \in N} R_i(s, a)$.

Foraging is a natural match to the revised model. The swarm seeks to maximize the sum of total number of pucks collected by the robots. Each robot should therefore try to maximize the number of pucks that were collected **overall** by *all individuals*, rather than individually.

The definition of the swarm utility and optimal strategy profile must now change appropriately:

$$
\begin{aligned}
U^T(\sigma) &= \sum_{t=1}^{T} R(s^t, \sigma(s^t)) \\
&:= \sum_{t=1}^{T} \sum_{i \in N} R_i(s^t, \sigma(s^t))
\end{aligned}
\tag{3.3}
$$

$$
\begin{aligned}
U(\sigma) &= \lim_{T \to \infty} \frac{1}{T} U^T(\sigma) \\
&= \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{i \in N} R_i(s^t, \sigma(s^t))
\end{aligned}
\tag{3.4}
$$

and

$$
\sigma^* := \arg\max_{\sigma} \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{i \in N} R_i(s^t, \sigma(s^t)) \tag{3.5}
$$

This change may seem innocuous but its closeness to reality undermines our ability to guarantee that a solution to the game would be stable (i.e., would be a Nash equilibrium). While the swarm seeks to maximize the joint reward R at any stage t, its members are **only aware** of their own individual payoffs R_i. Trivially, it would seem that selecting an action that maximizes R_i would also maximize R. But this is not the case, and Theorem 1 no longer holds.

We use the following example to illustrate. Let N be the set of agents, which are the members of a football team. Each agent has its **personal contribution**, which is the number of goals it scored. The team (swarm) profit function (R) describes the sum of individual contributions, which is the number of goals scored by all members of the team ($R = \sum_{i \in N} R_i$). Figure 3.4 illustrates a situation in which maximization of the team score is different from maximization of the individual contribution. Players p_1, p_2 are positioned such that p_1 holds the ball and needs to decide whether to try to score ('score' action) or to pass the ball to player p_2 ('pass' action). If it decides to score, it succeeds with a probability of 0.1 and both its personal contribution (which is the number of goals it scores) and the team profit increase, with the expected increase being 0.1 goals. Alternatively, if it decides to pass, player p_2 gets the ball and scores (with probability 1.0). In

this scenario, the personal contribution of player p_1 does not increase but the team profit increases by 1 with certainty.

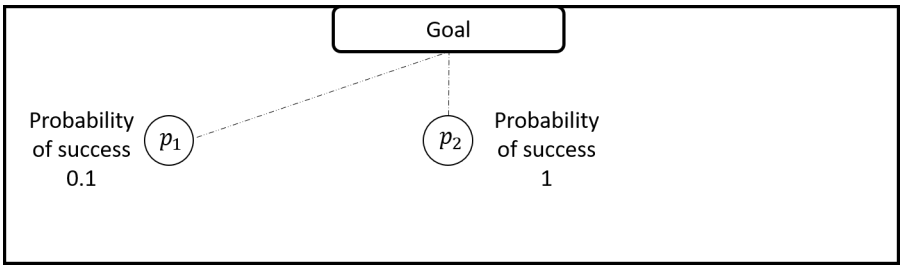

Figure 3.4: A situation in which the player has to choose between its own individual profit and the profit of the team.

If player p_1 knows the expected team rewards, it would undoubtedly choose the action 'pass', despite it having an individual reward of 0. However, as it only knows its own rewards, and thus will seek to maximize its own individual reward, taking the action 'score'.

This inherent difficulty in assessing the effects of an action on the welfare of the swarm, while having access (at best) only to an individual's own contribution (reward) is at the heart of the challenge faced by swarm members' action selection mechanism. It stems from ignorance: the individual swarm member does not know the expected individual rewards of other members, and thus cannot compute the expected sum of these rewards.

In swarm robots, ignorance of the individual swarm members is a result of several factors. Perhaps first among them is the computational difficulty, sometimes inability, of a robot to observe—and to recognize from the observations—the action and reward of another. For example in foraging, identifying that another robot is holding a puck may be computationally expensive (requiring recognition of a puck in an image), or even impossible (e.g., when the robot holding the puck is turned away such that its gripper is occluded). In the general case, distance to pucks is unavailable, and thus even simple reward estimates (those closest to picking up a puck should be given the right-of-way in a collision) are not feasible.

The reliance on abstract utilities whose measurement is impractical also limits the model in other ways. Even in foraging, where supposedly rewards are easily measured by pucks collected, it is difficulty to assess the value of collision avoidance action when no pucks are around, and the agents are unable to count how many pucks have been collected. Moving away from

foraging to other swarm tasks, the utility measure itself changes from one swarm task to another; there is no standard scale by which utility can be measured.

In Section 3.3 below, we take an important step towards removing one element in the ignorance of the agents with respect to their utility and those of others. We show that the swarm utility is improved by maximizing the **time** spent on the task, where time is of course a concrete, task-independent measure, which can be measured by all individuals in the swarm, at all times.

3.3 Using Time to Measure Utility

As discussed, the swarm seeks to maximize the long-term *mean* swarm utility defined above (Eq.3.4), but, under conditions where each member only knows its own reward R_i, it cannot determine the optimal response. Moreover, in reality, even this individual reward may be hidden from the robot itself, thus the social welfare game model is not directly applicable.

The first step we take to address this challenge of ignorance is by transforming the utility measure itself into a form directly measurable by every robot. This gives the robot a concrete measurable proxy to the utility, which they apply to evaluate their actions.

Similarly to previous investigations ([51, 32, 20]), we focus on *time* as a proxy to the utility gained by swarm members. In particular we consider the relation between the time spent by the swarm members on the task, and the utility resulting from it. We will prove that maximization of the first is equivalent to maximization of the second. And since the measurement of time can be carried out individually, without even knowing what the task is, the swarm robots can focus on increasing the time spent on the task, rather than maximizing some abstract notion of utility.

We remind the reader of two earlier points we established in earlier sections. First, that the swarm works in continuous time (Section 3.1.1): its members switch repeatedly between two modes of operation: a *program mode*, in which members carry out their individual tasks, and *collision mode*, in which members respond to collisions. Second, that the swarm utility U is given as the limit of means of the sum of individual payoffs (Section 3.2, Eq. 3.4). We will now combine these.

3.3.1 Individual *Program Mode* Duration as a Proxy to Individual Payoff

As a first step, we assume that each individual payoff (R_i) increase with time the agent is in working on its task (i.e., it is in program mode). Where the duration of the robot being in program mode is determined (as the reward) by a state and a joint action and is described by a function $P_i \colon S \times A \to \mathbb{R}$. This assumption is reasonable and based on two observations:

- First, robots are built to carry out their task during program mode, so it is reasonable to assume that whatever work they carry out, it is productive.
- Second, the robots coordinate when they are (in danger of) colliding, and it is therefore reasonable to expect that for the duration A_i of their resolving a collision (i.e., when they are in *avoidance* mode), they are not productive.

For simplicity, we assume that for any stage t, the individual payoff (R_i) increase proportionally **only** with the time spent on the task (P_i), i.e., $R_i \propto P_i$ (Assumption 5). Later, section 5.4 illustrates how this assumption plays a crucial role in our model, and how, when it fails, the guarantees are lost.

Assumption 5. *There exists $\beta \in \mathbb{R}^+$ such that $R_i(s^t, a^t) = \beta P_i(s^t, a^t)$, for all $i \in N, s^t \in S, a^t \in A$*

3.3.2 Total Direct Work of the Swarm

Given the relationship assumed above between the individual payoff and the duration of the program mode run-time, we can now re-define the global utility using the total accumulated duration of the program run-time of the individual robots.

We use the term *Direct Work* to refer to the total accumulated program time of the swarm members throughout the entire game, however many stages it takes (Definition 3.3.1). Direct work stands in contrast to *Overhead*, or *Indirect Work* (Definition 3.3.2), a term often used in business to denote expenses or activities which support and enable the direct work of a system. In the case of the swarm, the *overhead* of the system is the total time spent in coordination.

Definition 3.3.1. Direct Work
The *direct work* of a strategy profile σ is

$$P^T(\sigma) = \sum_{t=1}^{T} \sum_{i \in N} P_i(s^t, \sigma(s^t))$$

Definition 3.3.2. Overhead
The *overhead* of a strategy profile σ is

$$A^T(\sigma) = \sum_{t=1}^{T} \sum_{i \in N} A_i(s^t, \sigma(s^t))$$

We now move to demonstrate that the swarm utility of a strategy profile σ is directly tied to the direct work resulting from the same strategy (Lemma 1). Then, we show that maximizing the direct work also necessarily maximizes the utility (Theorem 2).

Lemma 1. *For every possible strategy σ, $U^T(\sigma) \propto P^T(\sigma)$.*

Proof.

$$\begin{aligned} U(\sigma) &= \sum_{t=1}^{T} \sum_{i \in N} R_i(s^t, \sigma(s^t)) && \text{[Eq. 3.1]} \\ &= \sum_{t=1}^{T} \sum_{i \in N} \beta P_i(s^t, \sigma(s^t)) && \text{[Assum. 5]} \\ &= \beta \sum_{t=1}^{T} \sum_{i \in N} P_i(s^t, \sigma(s^t)) \\ &= \beta P^T && \text{[Def. 3.3.1]} \end{aligned}$$

As $\beta \in \mathbb{R}+$, $U^T(\sigma) \propto P^T(\sigma)$. □

Based on Lemma 1 we can now show further that any strategy profile σ^* which maximizes P^T as T increases also maximizes U^T, i.e., it is an optimal strategy as defined in Def. 3.5.

Theorem 2. *Let σ^* be a strategy that maximizes P^T for any $T \geq 1$. Then σ^* is* an optimal strategy.

Proof. We are given σ^* maximizes P^T, for $T \geq 1$, i.e.,

$$\forall T, \sigma^* = \arg\max_{\sigma} P^T(\sigma)$$

Lemma 1 established $U^T(\sigma) \propto P^T(\sigma)$ for any σ, and therefore there exists a constant $\beta \in \mathbb{R}+$ such that $U^T(\sigma) = \beta P^T(\sigma)$. Thus, if σ^* maximizes P^T, then surely

$$\forall T, \sigma^* = \arg\max_{\sigma} \beta P^T(\sigma) = \arg\max_{\sigma} U^T(\sigma)$$

We get that the maximal value of U^T is $U^T(\sigma^*)$, i.e.,

$$\forall T, \forall \sigma, U^T(\sigma^*) \geq U^T(\sigma)$$

and as T is a positive integer, the following holds:

$$\forall T, \forall \sigma, \frac{U^T(\sigma^*)}{T} \geq \frac{U^T(\sigma)}{T}$$

and therefore,

$$\forall \sigma, \lim_{T\to\infty} \frac{U^T(\sigma^*)}{T} \geq \lim_{T\to\infty} \frac{U^T(\sigma)}{T}$$
$$\forall \sigma, \lim_{T\to\infty} \frac{1}{T} U^T(\sigma^*) \geq \lim_{T\to\infty} \frac{1}{T} U^T(\sigma)$$

Therefore, σ^* is an optimal strategy. □

We showed that maximization of the long-term swarm utility can be achieved by maximization of the direct work of the swarm, i.e., the total time spent by the swarm members in program mode. Taking only time into account, this function is independent of the task.

3.3.3 Revisiting the Unknown Horizon when Using Time

One of the constraints on any model of the swarm is that the duration of its total operation is unknown. In game-theoretic terms, this means that even if the number of stages in the game, T, is finite, it is unknown to the agents.

We have previously chosen to model this as the limit of means described in Definition 3.4, where the accumulating utility of the swarm is averaged over T, as $T \to \infty$.

This definition of the limit of means is satisfactory under the assumption that the duration of every stage is exactly the same, i.e, the swarm advances from one conflict (one stage game) to another in discrete, fixed-duration, steps. Alas, this is not the case in robotics.

In robot swarms, the horizon for the system is measured in *total operation time*, not in the number of collisions (which is the number of games taken, i.e., the number of collisions). When we deploy a swarm of robots, we are limited by their battery life, not by the number of collisions they face.

An example can illustrate intuitively. Suppose we have two joint avoidance actions available to the swarm members: action a resolves a collision in a minute; action b resolves a collision in 5 minutes. Suppose a first swarm, denoted Q_a, begins with a collision (everyone close together), and disperses to forage. Including the initial collision, it collides three times: each time it resolves the collision using action a, and continues to forage for 9 minutes after each collision. Thus the first stage has a $\sum_{i \in N} P_i()$ of $9 \times n$ minutes ($n = |N|$), and an avoidance taking one minute, the second stage similarly, etc. Overall, within 30 minutes, the n robots have worked for $3 \times 9 \times n$ minutes. Now consider a second swarm Q_b, that uses action b. Assuming the post-collision foraging time is still 9 minutes, we see that within 30 minutes, only two collisions could have taken place, each one lasting for $5+9 = 14$ minutes, for a total of 28 minutes. Thus within those 30 minutes, only $2 \times 9 \times n$ were spent foraging.

Fig. 3.5 illustrates how, eventually, swarm Q_a worked more time than Q_b even though both actions a and b have the same program time in each collision (and therefore, in average). Of course, post-collision foraging time may also vary, which reveals the importance of the **ratio** between program time and overall time.

The point is that averaging the foraging time on the basis of the number of collisions makes little sense when we seek to determine the productivity of a swarm. We should be averaging on the basis of the time spent in operation. We therefore revisit the formulation of the unknown horizon in the definition of the swarm utility. Rather than looking at the limit of means only the basis of the number of stages T, we also consider the time spent by the swarm in each stage, which is also referred to as the *length* of the stage, and described by $L\colon S \times A \to \mathbb{R}$. As with the other functions of this type, its

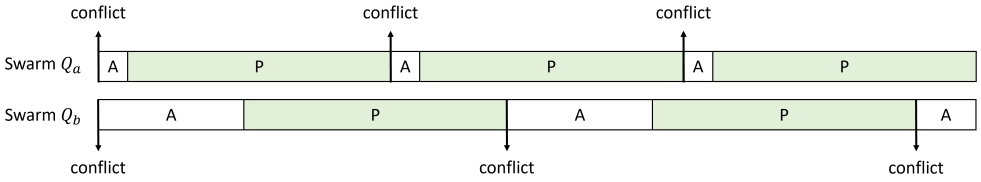

Figure 3.5: An illustration of the importance of collision time duration.A comparison of two swarms that differ in the actions they use, both actions achieve the same program time in each collision, but result in different long-term utilities. .

sum throughout all stages is denoted by $L^T(\sigma) = \sum_{t=1}^{T} L(s^t, \sigma(s^t))$.

Based on Lemma 1 and Thm 2, the original definition of the swarm utility (Definition 3.1.2) can be rewritten to use the direct work and L^T (Definition 3.3.3).

Definition 3.3.3. Time-based Utility

$$U(\sigma) := \lim_{T \to \infty} \frac{1}{L^T(\sigma)} U^T(\sigma)$$

We observe that L^T is simply the amount of time spent by the swarm in all T stages, collectively (i.e., the duration of operation of the swarm in the environment in which it is deployed). This can be computed directly.

The total time spent by the individual robots, is the sum of the direct work P^T (Def. 3.3.1) and the overhead A^T (Def. 3.3.2). These measure the total time of n robots spent in program mode, and avoidance mode, respectively, over all T stages of the game. Given n robots, who have been working for M minutes, the sum $P^T + A^T$ is $n \times M$. Diving this sum by the number of robots n will give the time spent by the swarm as a collective, i.e.,

$$L^T(\sigma) = \frac{P^T(\sigma) + A^T(\sigma)}{n} \tag{3.6}$$

Combining Def. 3.3.3 and 3.6 we can now rewrite the swarm utility using only time as the basis for the limit of means.

$$
\begin{aligned}
U(\sigma) &= \lim_{T\to\infty} \frac{1}{L^T(\sigma)} U^T(\sigma) && \text{[Def. 3.3.3]} \\
&= \lim_{T\to\infty} \frac{\beta}{L^T(\sigma)} P^T(\sigma) && \text{[Lemma 1]} \\
&= \beta \lim_{T\to\infty} \frac{P^T(\sigma)}{L^T(\sigma)} && \text{[}\beta\text{ is constant]} \\
&\propto \lim_{T\to\infty} \frac{P^T(\sigma)}{L^T(\sigma)} && \text{[}\beta > 0\text{]}
\end{aligned}
$$

3.4 Summary

The steps we have taken in this chapter show that a swarm of cooperative robots may coordinate to maximize the swarm utility, without directly measuring the utility itself. The swarm utility function is the sum of individual payoffs, which individuals cannot access. We show that instead, by relying on the assumption that utility increases with the time spent by robot swarm members directly working on the task, it is possible to indirectly improve the swarm utility, by increasing the total amount of time spent in direct work.

From theorem 1, we have learned that maximum payoff is also **stable** when applied to the situation, and from theorem 2, we have seen that the maximum payoff can be achieved only by considering time measurements. The combination of these two results allows us to conclude that the robots can use a time-based task-independent function to achieve a stable and optimal solution to any given task.

However, while the transition from abstract and task-dependent utility to time gives us hope that robots may be able to measure the swarm progress, the resulting function which we seek to maximize, given in Definition 3.3.3, cannot be directly computed in practice by robot swarm members. There are several reasons for this (which we address in the following chapter) but they all stem from the same underlying cause: the definition of the utility is given in swarm-centric terms, rather than individual-centric terms, i.e., the swarm utility is given in terms inaccessible to the swarm members.

Chapter 4

Swarming Bandits: A Prescriptive Model of Swarm Tasks

The previous chapter showed that an optimal strategy profile σ^* would maximize the expression (denoted hereafter by Ψ)

$$\Psi(\sigma) = \lim_{T\to\infty} \frac{P^T(\sigma)}{L^T(\sigma)}$$

This is a descriptive expression, from the point of view of the robots: it tells them what the swarm is seeking to maximize, but in terms they cannot control in practice. We now seek to transform this into a prescriptive model, that can be used in practice.

We begin by re-writing $\Psi(\sigma)$, so as to facilitate an robot-centric perspective. Expanding the utility expression above in greater detail (Definition 3.3.3), we see

$$\Psi(\sigma) = \lim_{T\to\infty} \frac{P^T(\sigma)}{L^T(\sigma)} \tag{4.1}$$

$$= \lim_{T\to\infty} \frac{\sum_{t=1}^{T} \sum_{i\in N} P_i(s^t, \sigma(s^t))}{L^T(\sigma)} \qquad \text{[Def. 3.3.1 in numerator]} \tag{4.2}$$

The denominator L^T is the duration of time in which the swarm was active, and it is the **same** for all robots, assuming all robots began deployment together. It can therefore be measured independently by each robot.

Maximizing Ψ by finding an optimal strategy profile σ^* faces two gaping challenges, which we address in the next sections:

- Given that any robot $x \in N$ does not know the transition probability function D, nor the reward function R_i (nor P_i nor A_i), how is it to choose its own action? We discuss online learning as a solution approach in Section 4.1.
- How does any robot $x \in N$ form its policy such that the swarm utility resulting from the joint action is maximized? We discuss a solution based on approximating each robot's marginal contribution in Section 4.2.

4.1 Learning a Strategy

The robot members of the swarm do not know what will be the direct work given a state and joint action, i.e., they do not know the function P. Thus seemingly cannot compute an optimal strategy maximizing Ψ. We propose that the robots learn these values using reinforcement learning.

A major challenge is that rewards are stochastic due to two reasons. First, the environment is stochastic in nature, this issue is described in greater detail in section 4.3.2. Second, the robots cannot communicate. They can only **partially influence** the joint action (by selecting an individual action that is part of the joint action) and are only **partially aware** of the joint action (they know their individual action). Consequently, robots can only learn according to their individual actions, leading to another type of stochasticity; robots may receive different rewards for the same individual action according to the choices made by others.

The lack of communication also prevents robot i from knowing the others' program times, meaning that it is able to measure P_i, but not P_j for $j \neq i$. We will ignore this issue for the remainder of this section, and return to it in section 4.2.

Let us furthermore assume for now that there is only one state, i.e., S is a singleton (with one single state, $|S| = 1$). The assumption will be relaxed later in this section.

We pose this as a *multi-arm bandit* (MAB) problem (also known as $k-$*armbandit* problem; see [57] for a recent introduction), which is designed to take into account stochastic rewards. Each robot is facing a choice of selecting between $k := |A_i|$ actions. Each time it makes a selection, it pays a price. The learning process attempts to learn as quickly as possible (i.e.,

with a minimal number of trials) the action that will generate the largest expected reward. In other words, the robot is trying to determine its own optimal individual action based on the rewards it observes after each one. To clarify, if we denote the single state by s and the set of stages in which player i played the individual action a_i as $T_i(a_i)$, then player i is looking for a_i^* such that:

$$a_i^* := \arg\max_{a_i} \lim_{T\to\infty} \frac{\sum_{t\in T_i(a_i)} P(s, a^t)}{\sum_{t\in T_i(a_i)} L(s, a^t)} \tag{4.3}$$

There are standard solutions to the MAB problem. Perhaps the most familiar of these is the Upper Confidence Bound (UCB1) algorithm [7]. The standard UCB1 algorithm ignores the duration of an action, just as in the standard definition of Markov Games, which ignores the duration of a stage. We therefore make a similar adjustment, as we have done adapting the formulation of the time-based utility (Def. 3.3.3 in Section 3.3).

Algorithm 1 below is a modified version of the UCB1 algorithm, adapted for taking the duration of each stage into account, to optimize a_i^* (Def. 4.3). The algorithm allows each robot to independently learn the value of each individual action a_i, denoted $Q(a_i)$. For brevity, in the algorithm description, the notation a refers to the **individual** action.

It should be noted that since UCB1 algorithm assumes a fixed (stationary) reward distribution, the second type of stochasticity that arises as a result of parallel learning may break its theoretical guarantees. The issue is discussed in more detail in section 5.3.2.

Multiple States and Contextual Bandits. We now turn to the more general case, where there are multiple states in the swarm game, i.e., $|S| > 1$. For example, in foraging, a collision from behind may be different from a collision up-front, both in the sense that different collision-avoidance actions may be available, and their rewards may greatly vary. Speeding up, for instance, may be a collision-avoidance method appropriate for near-collisions from behind, but inappropriate for head-on collisions.

Fortunately, the extension of multi-arm bandit problems for multiple states is quite familiar, and known as *contextual bandits* [36]. In contextual bandits, a separate $k - arm$ bandit exists for each separate state. The extension to Algorithm 1 is straightforward, and involves running a separate

Algorithm 1: Continuous-Time UCB Algorithm for Multi-Arm Bandits.

Initialize: $\forall a, Q(a) \leftarrow 0$ [Estimated value of action a.]
Initialize: $\forall a, count(a) \leftarrow 0$ [# of times a selected.]
Initialize: $\forall a, \sum P(a) \leftarrow 0$ [Accumulated reward for action a.]
Initialize: $\forall a, \sum L(a) \leftarrow 0$ [Accumulated stage duration when using a.]

1 **while** *true* **do**
2 Wait for a collision
3 Select action

$$a := \arg\max_a \Big[Q(a) + \sqrt{\frac{2 \ln(\sum_a count(a))}{count(a)}} \Big]$$

4 Execute a, observe A, P [Avoidance duration, Program duration.]
5 Increase $count(a) \leftarrow count(a) + 1$
6 Increase $\sum L(a) \leftarrow \sum L(a) + A + P$
7 Increase $\sum P(a) \leftarrow \sum P(a) + P$
8 Update $Q(a) = \frac{\sum P(a)}{\sum L(a)}$

update and action selection, with separate accumulators and counters, for each state. For brevity, we do not list the changes here.

Note that in contextual bandits, there is an assumption that contexts (states) are independent of each other and of the selected actions. The probability of encountering a state $s_x \in S$ is not related to the action taken in the previous stage, or any before it. This fits our assumption of uniform state transition probability (Assumption 4 in Chapter 3).

4.2 Lack of Knowledge of Others' Program Time

We have previously used Ψ to denote the swarm mean direct work

$$\Psi(\sigma) = \lim_{T\to\infty} \frac{\sum_{t=1}^{T}\sum_{i\in N} P_i(s^t, \sigma(s^t))}{\sum_{t=1}^{T} L(s^t, \sigma(s^t))}$$

. From the point of view of any robot x, this can be rewritten as:

$$\lim_{T\to\infty} \frac{1}{L^T(\sigma)} \sum_{t=1}^{T}\Big[P_x(s^t, \sigma(s^t)) + \sum_{i\neq x\in N} P_i(s^t, \sigma(s^t))\Big] \tag{4.4}$$

The term L^T is directly measurable by the robot (it is its own total deployment time), and of course the robot can also measure P_x for any t, and thus its own total time spent in program mode, within the total deployment time L^T. However, it cannot directly know how much time others have spent in their own program mode for the same stage, and therefore cannot access $\sum_{i\neq x\in N} P_i^t$.

Douchan *et al.* [20] encountered a similar challenge, and addressed it by circumventing direct computation of the total utility, replacing it with an approximation of the *marginal contribution* of the agent to the swarm utility. The marginal contribution can be described by the *Wonderful Life Utility* (WLU) function [65] which compares the swarm utility with the robot's contribution to the swarm utility if the robot is absent. (Def. 4.2.1).

Definition 4.2.1. The marginal contribution (WLU) of robot x to the total program time P in a given stage t is given by

$$\Delta^x P(s, a) := \Upsilon^x(s, a) - \Upsilon^{-x}(s, a) \tag{4.5}$$

where Υ^i is the total program time (in the stage t), when robot i is present, and analogously, Υ^{-i} is the total program time when robot i is absent from the stage.

Naively, one may therefore consider Υ^x to be the sum of swarm members' program mode duration (as before), and Υ^{-x} to be likewise, but with $P_x = 0$,

as robot x is absent, thus not in program mode. This leads to a trivial conclusion:

$$\begin{aligned}\Delta^x P(s,a) &= \overbrace{\sum_{i\in N} P_i(s,a)}^{\text{With } x} - \overbrace{[0 + \sum_{x\neq i\in N} P_i(s,a)]}^{\text{Without } x} \\ &= \big[P_x(s,a) + \sum_{x\neq i\in N} P_i(s,a)\big] - \sum_{x\neq i\in N} P_i(s,a) \\ &= P_x(s,a)\end{aligned}$$

However, this is not a correct definition. If the robot is absent, it does not take an action. And if it takes no action, then the rewards of the other robots would not be identical. In other words, when robot x is hypothetically absent, the time spent in program mode by any robot $i \neq x$, cannot be assumed to be the same as when x is present and participating in the game.

Before we dive into the question of how $\Upsilon^x, \Upsilon^{-x}$ should be defined, we first prove that the use of the WLU leads to optimality.

The remainder of this section will use the notation a_{-i} to describe the joint action a when the action of robot i is replaced with a dummy action which indicates its absence.

Theorem 3. *For each player i and for each state $s \in S$, maximization of the $\Delta^i U$ of some utility function $U\colon S \times A \to \mathbb{R}$ is equivalent to maximization of the utility function U.*

Proof. A robot that is trying to maximize the utility function U must select an optimal strategy given the strategies of the other agents. Consider its *order* of preferences: for an agent i at a cretin state $s \in S$, if the others' actions were given, which action should be preferred over another? The analysis below indicates that the order of preferences is the same for U and $\Delta^i U$, and therefore, we can consider the second rather than the former.

Let $a, b \in A$ be two joint actions that differ only in the action of player i. So, $a_{-i} = b_{-i}$.

We get that for any state $s \in S$:

$$\Delta^i U(s,a) - \Delta^i U(s,b) = \tag{4.6}$$
$$= \big[\Upsilon^i(s,a) - \Upsilon^{-i}(s,a)\big] - \big[\Upsilon^i(s,b) - \Upsilon^{-i}(s,b)\big] = \tag{4.7}$$
$$= \big[\Upsilon^i(s,a) - \Upsilon^i(s,b)\big] - \big[\Upsilon^{-i}(s,a) - \Upsilon^{-i}(s,b)\big] = \tag{4.8}$$
$$= \big[U(s,a) - U(s,b)\big] - \big[0\big] = \tag{4.9}$$
$$= U(s,a) - U(s,b) \tag{4.10}$$

As the difference is equal, we can conclude that the order of preferences according to U and according to $\Delta^i U$ is the same. □

4.2.1 Approximate Υ^x (when robot x is present).

Robot x knows its own P_x of course. But how is it to determine the duration of other robots' program mode duration? For any agent $i \neq x$, the robot x uses an estimate $\widetilde{P_i}$ instead of the actual value P_i which it does not know. As robot x only aware to its own choice, it uses the mean value it observed for its action in this state over the entire history as the estimator. Denote the set of stages in which the state was s and player x played the individual action a_x as $T_i(s, a_x)$ so the estimator, denoted by $\widetilde{P_x}$, will take the following form:

$$\widetilde{P_x}(s,a) = \frac{\sum_{t \in T_i(s,a_x)} P_i(s^t, a^t)}{|T_i(s,a_x)|}$$

As the other robots are indistinguishable to robot x, all P_i are estimated in the same way, by $\widetilde{P_x}$.

This results in estimating Υ^x, for any $s \in S, a \in A$, as follows:

$$\widetilde{\Upsilon^x}(s,a) := P_x(s,a) + \sum_{x \neq i \in N} \widetilde{P_i}(s,a) \tag{4.11}$$
$$= P_x(s,a) + (n-1)\widetilde{P_x}(s,a) \tag{4.12}$$

4.2.2 Approximate Υ^{-x} (when robot x is hypothetically not present).

Following the approach of Douchan, Wolf and Kaminka [20], we take it that when a robot is absent, it is unable to perform the task and therefore is considered to be in coordination mode during the entire cycle (Assumption 6).

Thus while indeed its $P_x = 0$, its A_x is maximal, and equal to the entire duration of the stage t, which we have previously denoted by L (Section 3.3.3).

Recall that a_{-i} is used to describe the joint action a when robot i is absent.

Assumption 6. *When agent i is absent, it is considered to be in coordination mode for the entire cycle.*

$$\forall i \in N, \forall a \in A, \forall s \in S:$$

$$P_i(s, a_{-i}) = 0 \Rightarrow A_i(s, a_{-i}) = L(s, a) \tag{4.13}$$

This allows agent x to define its own P_x and A_x when it is absent. But what about the other robots?

We assume that when the robot x is absent, the collision can be treated as if it had never happened, and thus the other robots' program mode would not be interrupted by a collision.

Figure 4.1 demonstrates this hypothetical timeline. The top figure shows the actual time line. The middle figure considers the case where there was no collision, as the robot in question was absent and no collision occurs, thus the conflict and its avoidance duration should not exist. The bottom figure shows a revised hypothetical timeline, whereby the other robots where not interrupted by a collision, and thus enjoy two periods of program mode.

For each robot $i \neq x$, the approximation of P_i when robot x is absent, is defined to be $(2 \times \widetilde{P_x})$, where $\widetilde{P_x}$ is the mean program time robot x observed after applying its individual action in the state s, as discussed in Section 4.2.1.

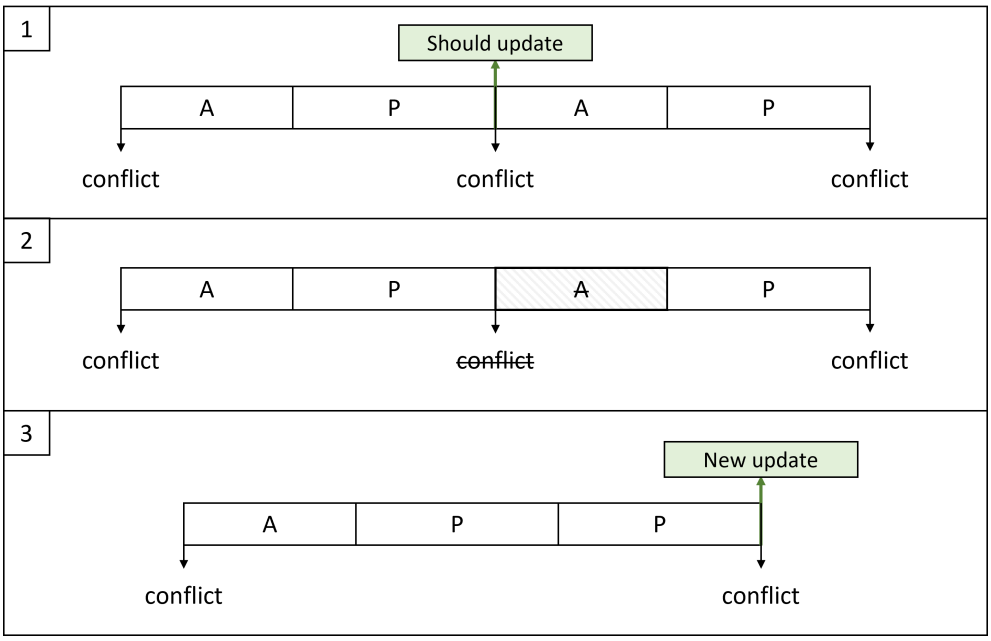

Figure 4.1: The hypothetical timeline when collision is treated as if it had never happened.

Taking these together, Υ^{-x} is approximated as

$$\widetilde{\Upsilon^{-x}}(s,a) := \sum_{i \in N} \widetilde{P}_i(s, a_{-x}) \tag{4.14}$$

$$= \widetilde{P_x}(s, a_{-x}) + \sum_{x \neq i \in N} \widetilde{P}_i(s, a_{-x}) \tag{4.15}$$

$$= 0 + \sum_{x \neq i \in N} \widetilde{P}_i(s, a_{-x}) \qquad \text{[Assumption 6]} \tag{4.16}$$

$$= 0 + \sum_{x \neq i \in N} 2\widetilde{P_x}(s,a) \qquad \text{[]Discussion above.]} \tag{4.17}$$

$$= 2 \sum_{x \neq i \in N} \widetilde{P_x}(s,a) \tag{4.18}$$

$$= 2(n-1)\widetilde{P_x}(s,a) \qquad \text{[Same estimate for all robots.]} \tag{4.19}$$

4.2.3 Estimated WLU

Taking Υ^x from Section 4.2.1 and Υ^{-x} from Section 4.2.2 together, we can now define robot x's estimate of its marginal contribution to the total program time, given a state s and a joint action a, with the WLU as:

$$\widetilde{\Delta^x P(s,a)} := \widetilde{\Upsilon^x}(s,a) - \widetilde{\Upsilon^{-x}}(s,a) \tag{4.20}$$

$$= P_x(s,a) + (n-1)\widetilde{P_x}(s,a) - \Upsilon^{-x}(s,a) \qquad \text{[Eq. 4.12]} \tag{4.21}$$

$$= P_x(s,a) + (n-1)\widetilde{P_x}(s,a) - 2(n-1)\widetilde{P_x}(s,a) \qquad \text{[Eq. 4.19]} \tag{4.22}$$

$$= P_x(s,a) - (n-1)\widetilde{P_x}(s,a) \tag{4.23}$$

Estimating the Size of the Swarm. The estimated $\widetilde{\Delta^x P}$ relies on knowing the **size** of the swarm n. Indeed, We have so far assumed that all robots participate in every collision (Assumption 2), and thus the estimation uses n (actually, $n-1$) in its computation.

However, individual robots do not know the size of the swarm. We follow Douchan, Wolf, and Kaminka [20] and use instead the number of **affected**

robots participating in the collision, which we denote by n_0. n_0 is approximated by counting the number of sensors that are blocked (i.e., detect a robot in collision-distance). Figure 4.2 illustrates a number of scenarios and the corresponding n_0 approximations: (a) a collision of two robots, each recognizing that the collision is with one robot exactly; (b) a collision of two robots, with one underestimating n_0; (c) a collision of two robots, with one overestimating n_0; (d) a collision of three robots in which one is aware of all others, one is partially aware, and one is not aware at all; (e) a collision of three robots, with one overestimating n_0; and (f) a collision of five robots, with some of them underestimating and some overestimating.

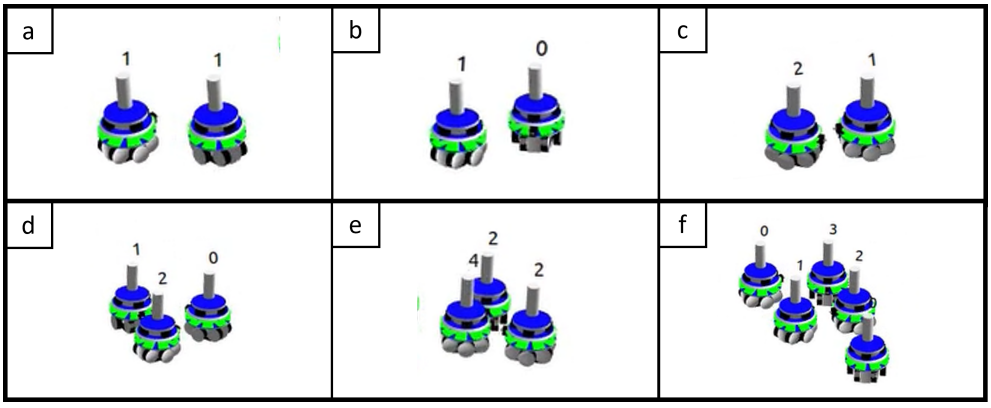

Figure 4.2: Examples of n_0 approximation as captured by the experimental simulator.

Algorithm 2 below describes the single robot learning procedure, with all modifications, as it was used during the experiment phase (in Chapter 5). As this learning algorithm executes in each state separately, as appropriate for assumption 4, we ignore the state notation in the description of the algorithm. Also, as before, the notation a refers to the individual action.

Algorithm 2: Continuous-Time UCB Algorithm for Swarm Robotic Tasks.

Initialize: $\forall a, Q(a) \leftarrow 0$ [Estimated value of action a.]
Initialize: $\forall a, count(a) \leftarrow 0$ [# of times a selected.]
Initialize: $\forall a, \sum P(a) \leftarrow 0$[Accumulated program time for action a.]
Initialize: $\forall a, \sum L(a) \leftarrow 0$ [Accumulated stage duration when using a.]
Initialize: $\forall a, \sum R(a) \leftarrow 0$ [Accumulated reward for action a.]

1 **while** *true* **do**
2 Wait for a collision
3 Observe n_0 [Number Of robots affected.]
4 Select action

$$a := \arg\max_a \Big[Q(a) + \sqrt{\frac{2\ln(\sum_a count(a))}{count(a)}} \Big]$$

5 Execute a, observe A, P [Avoidance duration, Program duration.]
6 Increase $count(a) \leftarrow count(a) + 1$
7 Increase $\sum L(a) \leftarrow \sum L(a) + A + P$
8 Increase $\sum P(a) \leftarrow \sum P(a) + P$
9 Set $R = P - n_0 \times \frac{\sum P(a)}{count(a)}$
10 Increase $\sum R(a) \leftarrow \sum R(a) + R$
11 Update $Q(a) = \frac{\sum R(a)}{\sum L(a)}$

4.3 This Model Generalizes Previous Work

We have referred throughout the exposition to earlier closely related work by Kaminka, Erusalimchik and Kraus (KEK, [32]), and by Douchan, Wolf, and Kaminka (DWK, [20]). Both of these are special cases of the model presented above, with the exception that they may be utilizing different approximation methods (which we empirically evaluate in Chapter 5). We discuss the generalization of these earlier models in this section.

4.3.1 The Assumption of a Stationary Strategy

Both KEK and DWK models begin their modeling process by considering an extensive form game, in which every conflict is a decision node. This allows a description of the swarm's evolution with time, such that the history of decisions influences current decisions. They then introduce an assumption that history does not matter to the winning strategy, and thus every decision-node should be considered a normal-form game independent of previous ones, i.e., they flatten the extensive-form game into a series of repeated games. Repeated games [9] are special case of stochastic games [55], where $|S| = 1$.

We make the same assumption here, in that we consider only stationary strategy profiles. However, while KEK and DWK assume there is only one type of conflict that could occur we do not make this assumption. We offer a model that allows multiple conflict states, though we make the assumption that the transition probability between states is uniformly distributed.

4.3.2 Stochastic Rewards

One of the key assumptions of both KEK and DWK models is that if the same joint action is performed by the agents to resolve the conflict, the consequences will be the same, regardless of **when** it occurs. In practice, this does not hold. There are multiple reasons for this, but the fact is that in practice, the duration of program mode and the duration of collision avoidance can vary quite a bit.

As an example, consider a foraging task perform by a swarm of two agents, where when they collide one turns left for two seconds and the other turns right for two seconds. Two situations are shown in figure 4.3, illustrating how the results of this joint action can be different. On the above image, both agents are heading for the nest while colliding, once they have finished

coordinating, they return to performing their task and must first fix their direction. This causes them to collide again, which ends the cycle and rewards both of them with 0.5 - for two seconds in avoid time and two seconds in program time. In contrast, in the bellow image, the agents are searching for food while colliding, so when they return to perform their task, they just keep moving forward. The reward in this scenario will be higher than 0.5 because it will take them more time to collide again.

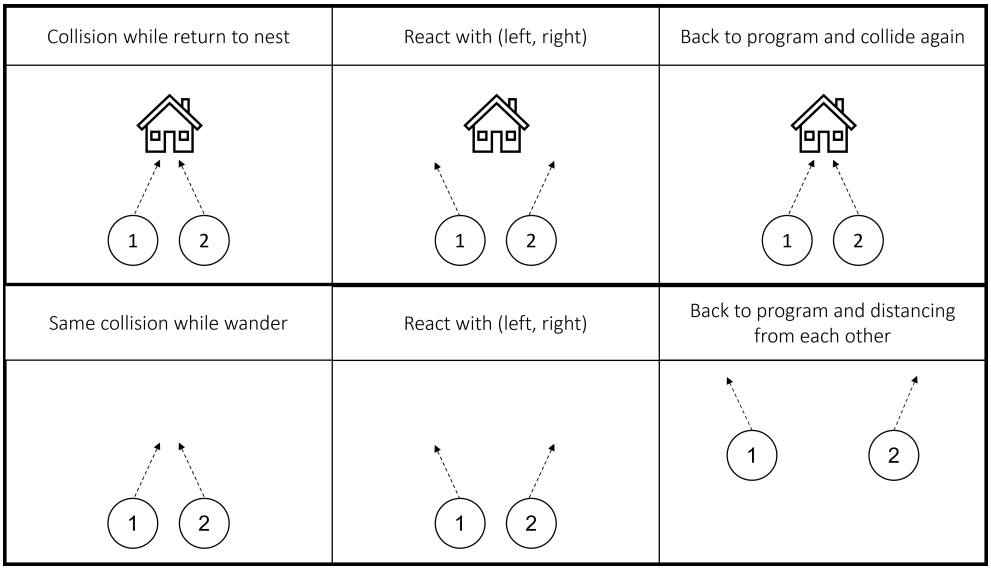

Figure 4.3: An example of how the world state can affect the consequence

DWK report on this [20] but do not address this in the formulation of the model. Instead, they attempt to address the stochastic nature in practice by an ad-hoc modification to the learning algorithm to handle this variance, while allowing convergence. A different approach was taken by KEK [32], where the experiments used a high learning rate (0.5) with a standard Q-learning algorithm, to re-adapt to new rewards and quickly ignore older ones, sacrificing convergence to an optimal action.

In contrast, our formulation of the learning problem as a multi-arm bandits easily and elegantly admits stochastic rewards, which means we can use standard algorithms for learning. The only change in the algorithm is to admit actions with duration.

The key difference between KEK/DWK and the formulation presented in this paper is that both KEK and DWK attempted to determine the mean

overhead (termed EI), which is the ratio of the avoidance duration to the total stage duration, for any given stage t:

$$Q(a) := \mathbb{E}[\frac{A_i(a)}{(A_i(a) + P_i(a))}]$$

When this is averaged over the lifespan over the robot, KEK and DWK are averaging the ratio. Instead, the formulation we present in this paper considers only the direct work P_i:

$$Q(a) := \mathbb{E}[P_i(a)]$$

The formulation as a multiarm bandit problem admits stochastic rewards much more readily.

4.3.3 Direct Work vs Indirect Work (Overhead)

DWK define the *Coordination Overhead* (CO) of the system, which is analogous (but inverse) to the notion of direct work which we utilize. We show this below.

The CO of the swarm, given a strategy σ is defined by DWK (Def. 3.1 in [20])as follows[1]:

$$CO(\sigma) := \frac{1}{L^T(\sigma)} \sum_{i \in N} \sum_{j=1}^{T} A_i(s^t, \sigma(s^t))$$

DWK assumed T was given, but unknown to the robots. They did not recognize the infinite horizon settings formally, and so did not present the limit of means notation as we do. However, the underlying mechanism of learning and the mathematical derivations continue to assume T is not known. So in actuality, a more correct formulation of CO would be:

$$CO(\sigma) := \lim_{T \to \infty} \frac{1}{L^T(\sigma)} \sum_{i \in N} \sum_{j=1}^{T} A_i(s^t, \sigma(s^t)) \tag{4.24}$$

[1]We converted their notations to those used in this paper

It is easy to see that this is quite similar to our definition of $\Psi(\sigma)$ in Eq. 4.2):

$$\Psi(\sigma) = \lim_{T\to\infty} \frac{1}{L^T(\sigma)} \sum_{t=1}^{T} \sum_{i\in N} P_i(s^t, \sigma(s^t))$$

We now proceed to show that in fact the two definitions are inversely related: maximizing Ψ is equivalent to minimizing CO, and vice versa. To do this, we first define a positive-linear relationship between two functions and denote it as $\overset{+}{\sim}$ (Definition 4.3.1).

Definition 4.3.1. Positive-Linear Relationship
A function $f(x)\colon \mathbb{R} \to \mathbb{R}$ is positive-linear to the function $g(x)\colon \mathbb{R} \to \mathbb{R}$ if and only if there exists $a \in \mathbb{R}^+$ and $b \in \mathbb{R}$ such that $\forall x \in \mathbb{R}\colon f(x) = ag(x)+b$
This relationship is denote as $f(x) \overset{+}{\sim} g(x)$

One characteristic of this relationship is that maximization of one function is equivalent to maximization of the other (Lemma 2).

Lemma 2. *If $f(x)$ is positive-linear to $g(x)$ then maximization of $f(x)$ is equivalent to maximization of $g(x)$.*

Proof. From Def. 4.3.1, we know $\exists a \in \mathbb{R}^+$ and $\exists b \in \mathbb{R}$ such that: $f(x) = ag(x) + b$. Then,

$$\begin{aligned} f(x) = ag(x) + b &\iff (f(x))' = (ag(x) + b)' && \text{[Differentiate both sides.]} \\ &\iff f'(x) = ag'(x) + 0 && \text{[Differentiation rules.]} \end{aligned}$$

Since $a \in \mathbb{R}^+$ the extrema points will be at the same value of x. □

Using Lemma 2 we can now relate Ψ and CO, and show that maximizing one, minimizes the other (Theorem 4).

Theorem 4. *Given a strategy profile σ for a swarm game G, maximizing $\Psi(\sigma)$ is equivalent to minimizing $CO(\sigma)$.*

Proof.

$$
\begin{aligned}
\Psi(\sigma) + CO(\sigma) &= \overbrace{\lim_{T\to\infty} \frac{1}{L^T(\sigma)} \sum_{t=1}^{T} \sum_{i\in N} P_i(s^t, \sigma(s^t))}^{\Psi} + CO(\sigma) && \text{[Eq. 4.2]} \\
&= \lim_{T\to\infty} \frac{1}{L^T(\sigma)} P^T(\sigma) + CO(\sigma) && \text{[Def. 3.3.1]} \\
&= \lim_{T\to\infty} \frac{1}{L^T(\sigma)} P^T(\sigma) + \overbrace{\lim_{T\to\infty} \frac{1}{L^T(\sigma)} \sum_{t=1}^{T} \sum_{i\in N} A_i(s^t, \sigma(s^t))}^{CO} && \text{[Eq. 4.24]} \\
&= \lim_{T\to\infty} \frac{1}{L^T(\sigma)} P^T(\sigma) + \lim_{T\to\infty} \frac{1}{L^T(\sigma)} A^T(\sigma) && \text{[Def. 3.3.2]} \\
&= \lim_{T\to\infty} \frac{1}{L^T(\sigma)} [P^t(\sigma) + A^T(\sigma)] \\
&= \lim_{T\to\infty} \frac{1}{\frac{[P^T(\sigma)+A^T(\sigma)]}{n}} [P^t(\sigma) + A^T(\sigma)] && \text{[Eq. 3.6]} \\
&= \lim_{T\to\infty} \frac{n[P^T(\sigma) + A^T(\sigma)]}{[P^t(\sigma) + A^T(\sigma)]} \\
&= \lim_{T\to\infty} n && [[P^T + A^T] \text{ cancel}] \\
&= n
\end{aligned}
$$

It follows that $\Psi(\sigma) = -CO(\sigma)+n$, and as n is constant, $\Psi(\sigma) \overset{+}{\sim} -CO(\sigma)$ (Definition 4.3.1). Then, based on Lemma 2, we conclude that maximizing $\Psi(\sigma)$ will maximize $[-CO(\sigma)]$, i.e., minimize $CO(\sigma)$. □

Chapter 5

Experiments

We now turn to empirically evaluate the model and learning approach presented in Chapter 4. The experiments' settings are described in Section 5.1.

5.1 Setup: The Krembot Simulator

The experiments were carried out on a simulator called the *krembot-sim*, available online at `https://bitbucket.org/galk-opensource/krembot_sim`. The simulator is based on the *Argos simulator* [46], which is often used in swarm robotics research. Krembot-sim simulates *krembot* robots in our lab[1], shown in Figure 5.1. These are small and simple robots that designed for swarms research. The Krembots are round, with a diameter of 6.5 centimeters and height of 10.6 centimeters. They have 8 visual and range sensors spread uniformly around the robot circumference, allowing the distinction of color and distance in 8 directions.

5.1.1 Physical Environment Settings

We have constructed several settings for foraging. Figure 5.2 shows the arena we used, it is 1 square simulated meter. The home is in the center of the arena; its radius is 0.1 meters. pucks are concentrated in bases, which are scattered in fixed locations (see below). Each such base is a circle 0.03 meters in radius.

[1]More documentation about the krembots can be found at: `https://robotican.net/krembot`

Figure 5.1: The krembots in our lab.

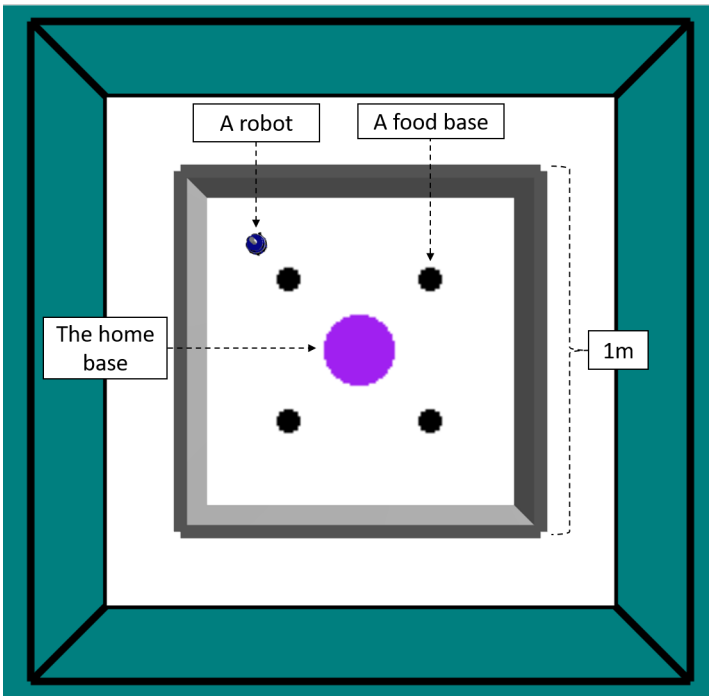

Figure 5.2: The Arena

We used two different settings in determining the location of puck bases:

Fixed food locations. There are four food bases, each is located 0.3 meters from the center of the nest as shown in Figure 5.2.

Random food locations. The food locations are randomized from a uniform distribution over the arena such that we restrict the locations inside the nest or adjacent to obstacles (the walls). A new food location is randomized every time a robot **picks up** food, i.e, at any time there are four available food locations in the arena.

We tested five swarm sizes in the same arena (in essence, varying the density): 5, 11, 17, 23 and 30 robots.

5.1.2 The Foraging Algorithm

The foraging algorithm itself is composed of two modes, as described briefly above: a task mode (program mode), and a coordination mode (collision avoidance mode), which is triggered when a collision is imminent.

Foraging Task Mode (Program Mode). Two behaviors were developed, *wander* and *return to nest*. The former is used when the robot has no puck, and is searching for one. It switches to the latter when a puck is found (i.e., the robot's position is within a puck base). The two behaviors are described below.

Wander. The agent is looking for pucks. It goes straight until it senses an obstacle and then turns 135 degrees back into the arena (Figure 5.3).

Return To Nest. The agent has a puck and needs to get to the home base to put it there. It does this by **navigation**. The agent is provided with two pieces of information in order to perform the navigation behavior: its own position and angle relative to the X axis, and the location of the home base. Given its own position and the location of the home base, it calculates the angle between the line connecting these two points and the X axis. Based on this angle and its own current angle, it turns toward the home base. Then, it goes straight until it reaches its destination, except if it collides with other robots along the way, in which case it performs afterward the "angle correction" procedure described above once again.

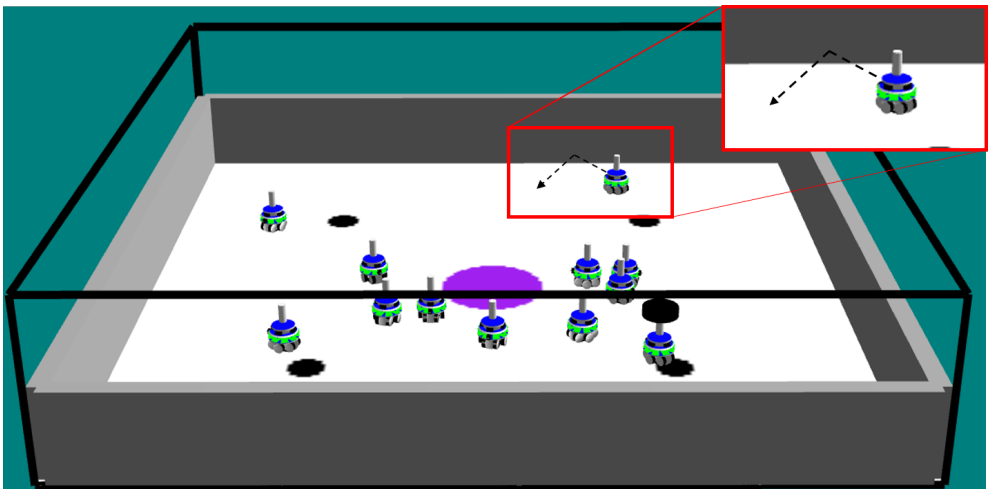

Figure 5.3: Throw back inside behavior

Foraging Coordination Mode (Avoidance Mode). For the remainder of this explanation, the sensor names described in Figure 5.4 will be used. We say that there is a robot in front of a sensor if this sensor detects an object at a distance of 4 centimeters while at the same time detecting a green light (during the experiments, all robot's LEDs are lit green to enable them to distinguish between one another and objects). A **conflict** occurs when a robot senses another robot from its Front, FrontLeft or FrontRight sensors (considers as a front collision) or its Rear, RearLeft or RearRight sensors (considers as a back collision). In many experiments, only a single avoidance state was used (in particular in comparison with related work, which relied on a repeated game model of swarm behavior), but in some cases (denoted *multistate* in the presentation of results), the front and back collisions were separately addressed, each with its own learning mechanism and counters.

We tested several sets of collision-avoidance methods, both using the *repeal* behavior with different intervals of execution. In this behavior, the robot attempts to go in the direction **opposite** the collision direction for half of the interval, then stop for the remainder. Note that, the purpose of the stopping part is to reduce the impact of the behavior on the task execution. A more detailed discussion of this issue can be found in Section 5.4.2.

Two sets of actions were used, shown in Table 5.1.

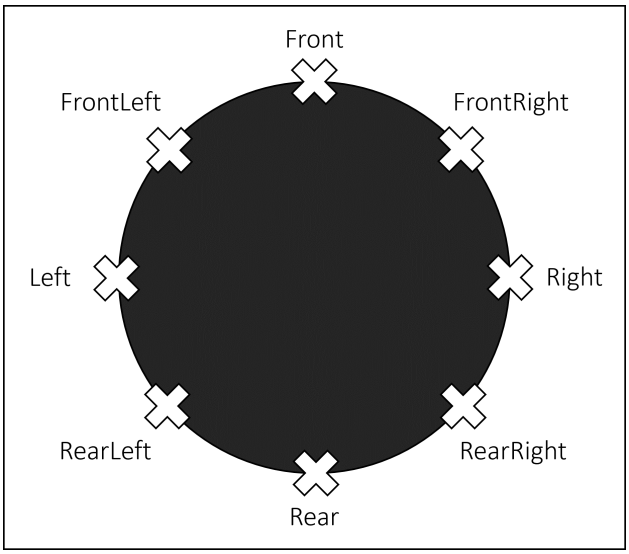

Figure 5.4: The Krembot sensors' names

Avoid action set	Repeal interval (milliseconds)	No. of Actions
1	500, 1000	2
2	200, 500, 1000	3

Table 5.1: Two sets of avoidance actions used in the experiments.

5.1.3 Foraging Performance Measurement

Each experiment with a specific setting (selection algorithm, puck base locations, size of swarm, etc.) goes through a learning episode followed by a testing episode. When using a non-learning algorithm (i.e., an algorithm that always chooses a specific method), the learning episode is skipped.

In the learning episode, the swarm is operated for 8 simulated hours, to allow ample time for each robot to converge to its preferred stationary strategy. At the end of the learning episode, we record for each robot its optimal action (maximizes $Q(a_i)$), and a strategy profile is generated by joining together the stationary strategy of each individual. This was repeated five (5) times, with five different random seeks.

The resulting five strategy profile were used in a testing episode: 50 experiments of 20 simulated minutes, each with a different random seed. For the non-learning algorithms, 50 such experiments ran for each of the fixed strategies (selection algorithms).

Overall, the main measure is the average number of pucks that the agents collect during the testing episode. For each of the fixed selection algorithms, the results are over 50 experiments. For the learning algorithms, the results are over 250 experiments (5 learning episodes, 50 testing episodes to each of them).

5.2 Swarming Bandits Compared to Fixed Policies

This section presents the results of experiments carried out in comparing the swarming bandits model (Chapter 4) to swarms where all robots are homogeneous in their decision-making: they all select the same avoidance action whenever the need for coordination arises. We contrast two version of the swarming bandits model: one with a single-state (i.e., a repeating games version, called *stateless* below), and one with two states (differentiating contexts based on front and back collisions, called *multistate*).

All the figures have the same structure: The vertical axis denotes the algorithms used in selecting avoidance actions. Algorithms with names that have the pattern *fix-** are those with fixed avoidance methods. For example, *fix-repelOpposite500* refers to the settings where all robots used repel, where the interval was set to 500ms. The horizontal axis measures the average number of pucks collected during the testing episodes (as discussed above). The results of all runs of a specific algorithm are displayed as a horizontal boxplot.

Each figure below shows the results from trials with a given number of robots n, a given set of avoidance actions (*Set 1* or *Set 2*), and puck base location settings (*Fixed* locations, or *Random*).

5.2.1 Set 1: Fixed Puck Base Locations

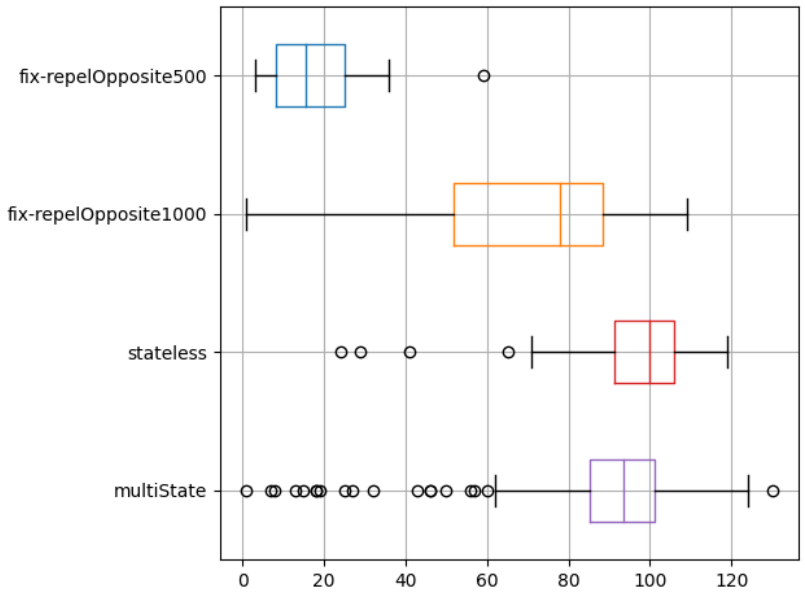

Figure 5.5: 5 Robots - Set 1 - Fixed Food Locations

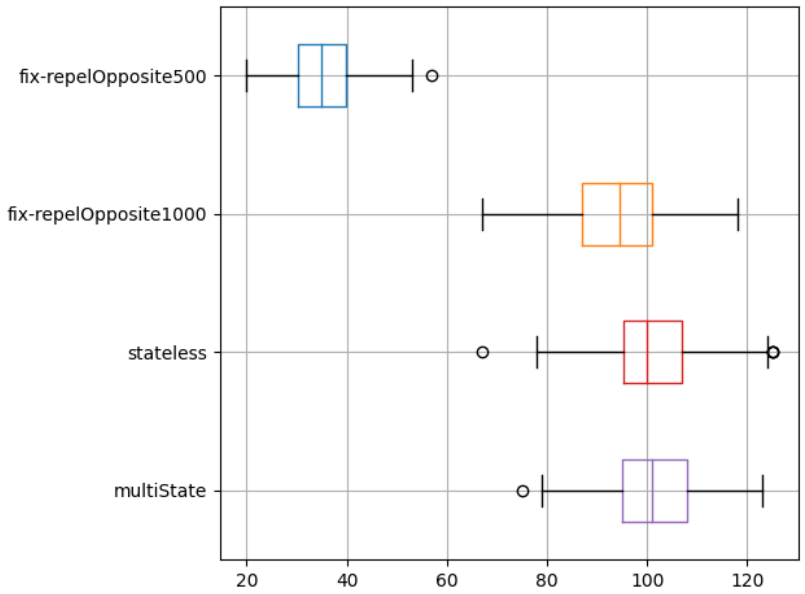

Figure 5.6: 11 Robots - Set 1 - Fixed Food Locations

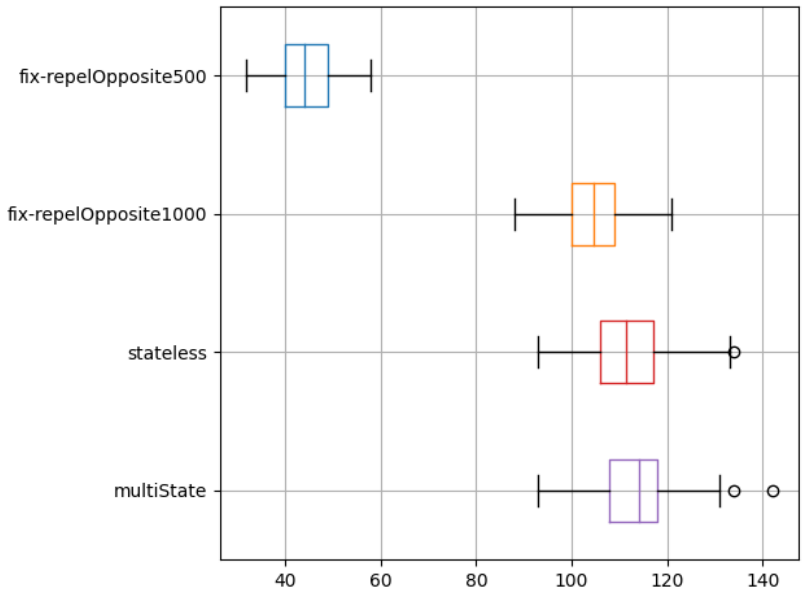

Figure 5.7: 17 Robots - Set 1 - Fixed Food Locations

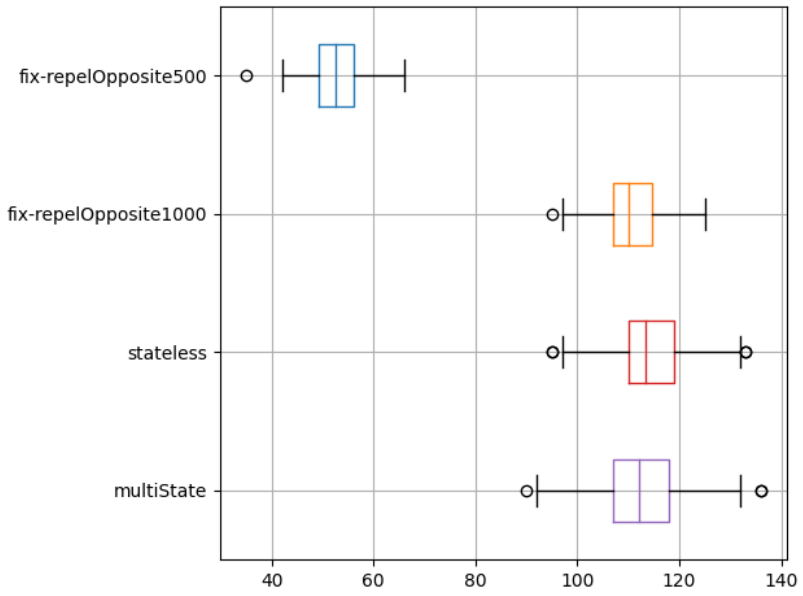

Figure 5.8: 23 Robots - Set 1 - Fixed Food Locations

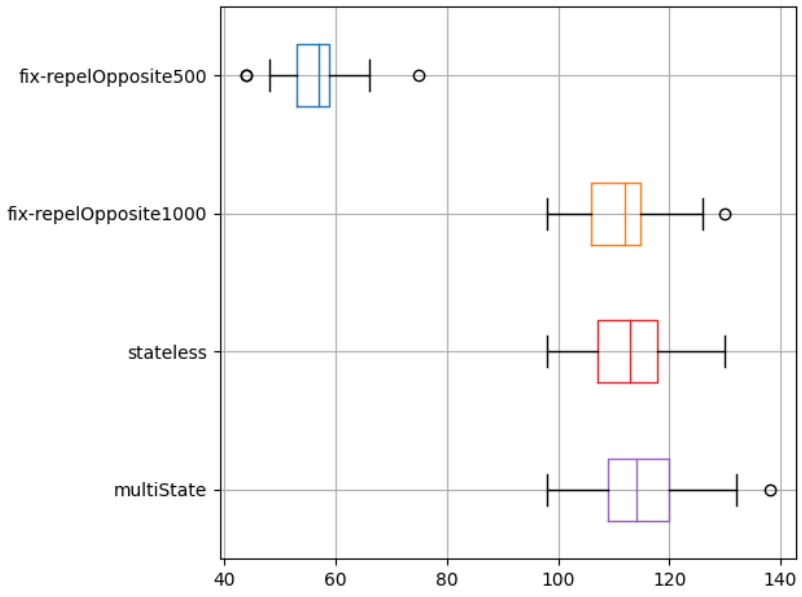

Figure 5.9: 30 Robots - Set 1 - Fixed Food Locations

5.2.2 Set 1: Random Puck Base Locations

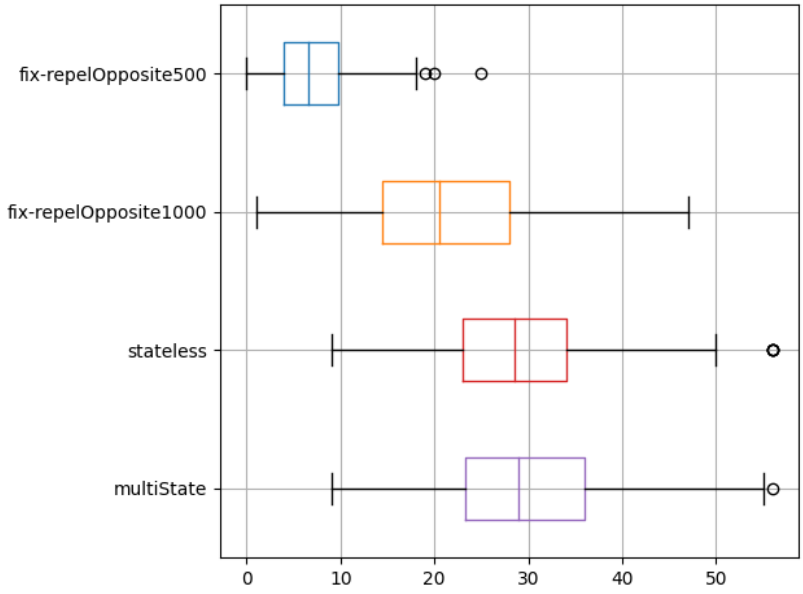

Figure 5.10: 5 Robots - Set 1 - Random Food Locations

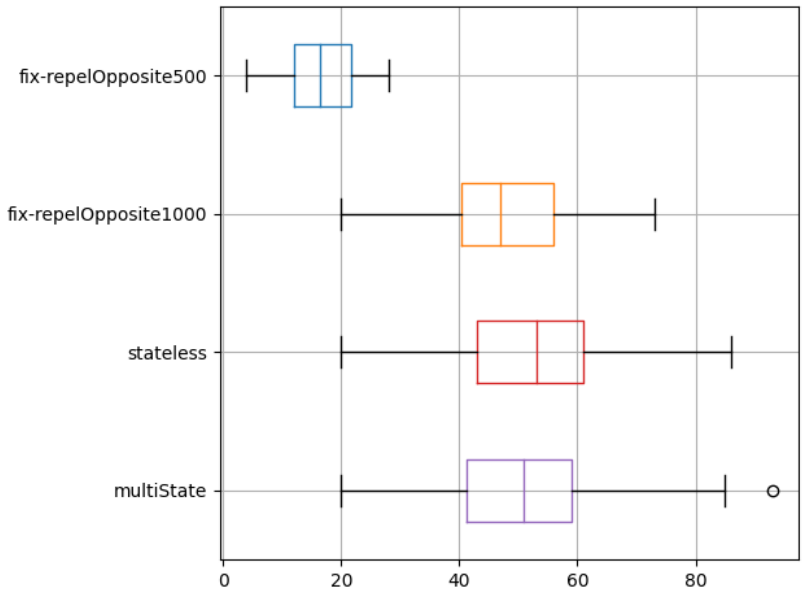

Figure 5.11: 11 Robots - Set 1 - Random Food Locations

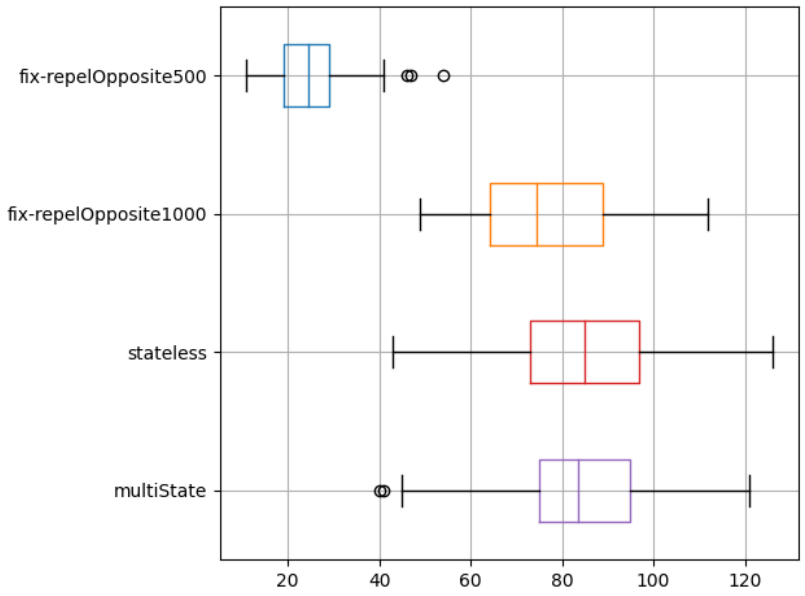

Figure 5.12: 17 Robots - Set 1 - Random Food Locations

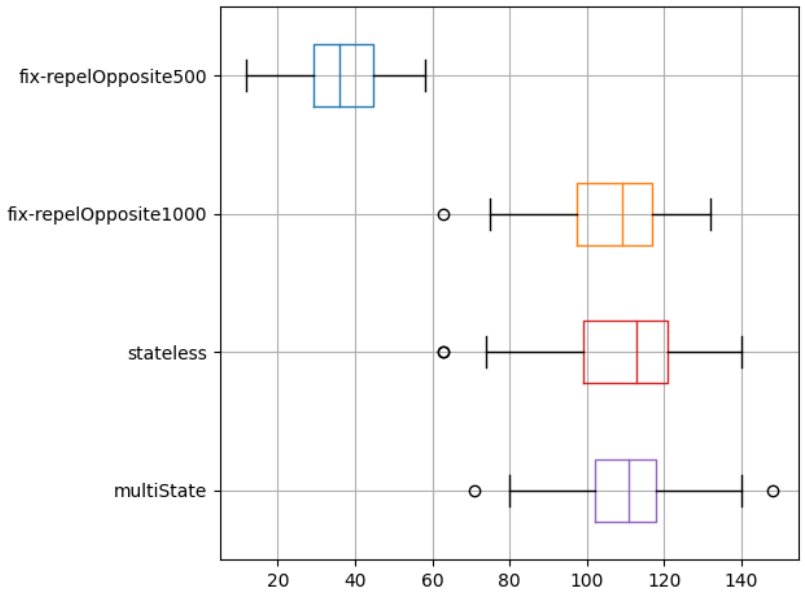

Figure 5.13: 23 Robots - Set 1 - Random Food Locations

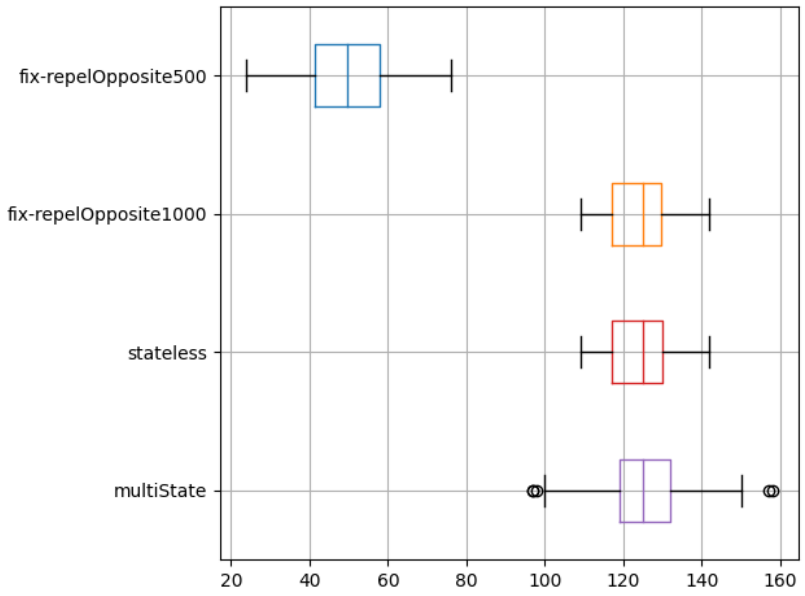

Figure 5.14: 30 Robots - Set 1 - Random Food Locations

5.2.3 Set 2: Fixed Puck Base Locations

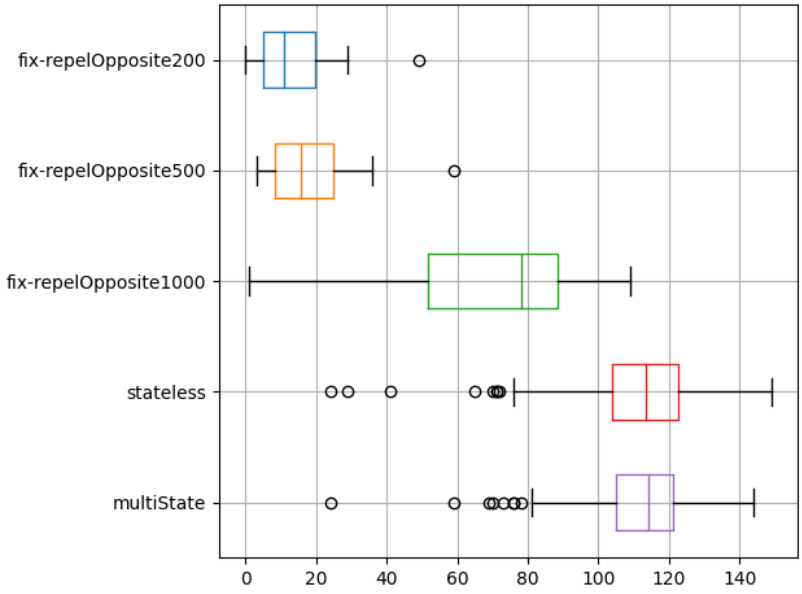

Figure 5.15: 5 Robots - Set 2 - Fixed Food Locations

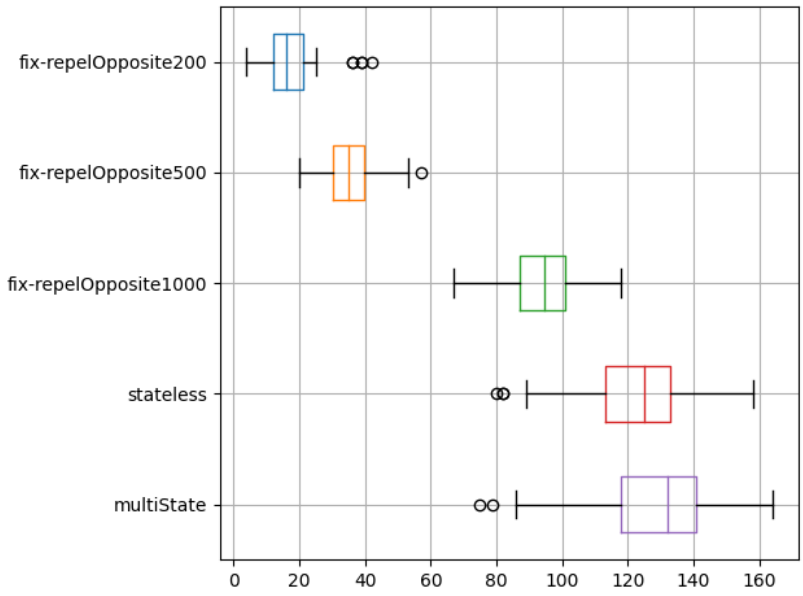

Figure 5.16: 11 Robots - Set 2 - Fixed Food Locations

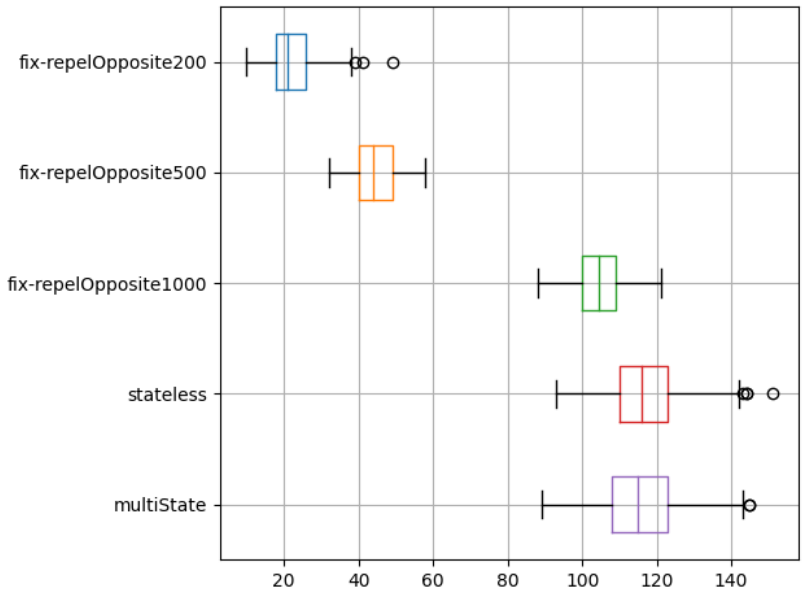

Figure 5.17: 17 Robots - Set 2 - Fixed Food Locations

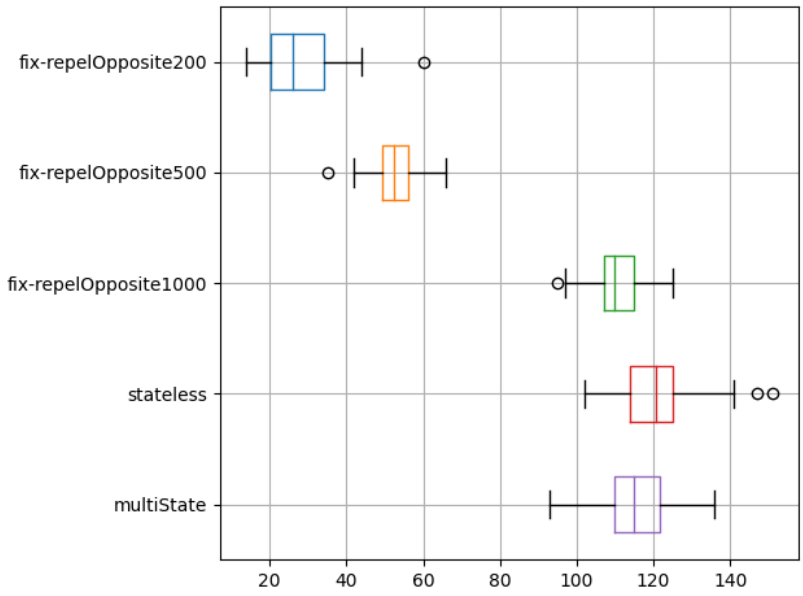

Figure 5.18: 23 Robots - Set 2 - Fixed Food Locations

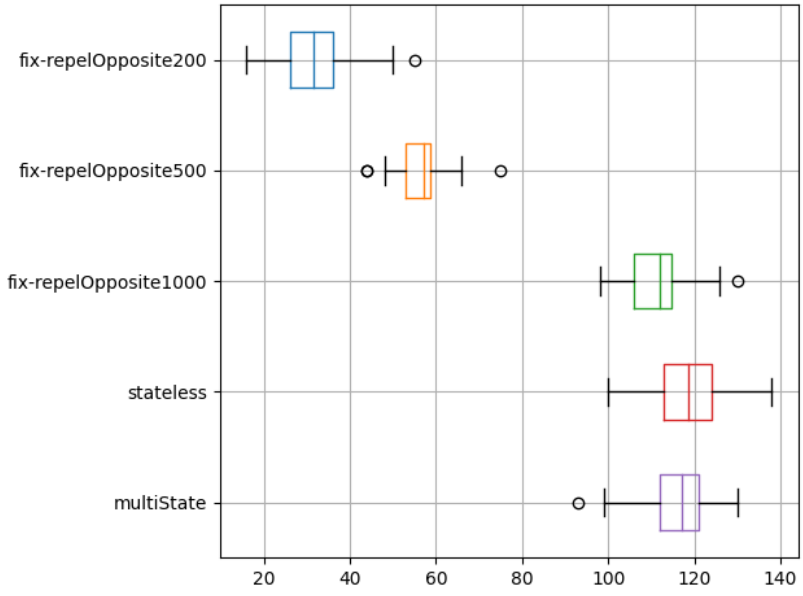

Figure 5.19: 30 Robots - Set 2 - Fixed Food Locations

5.2.4 Set 2: Random Puck Base Locations

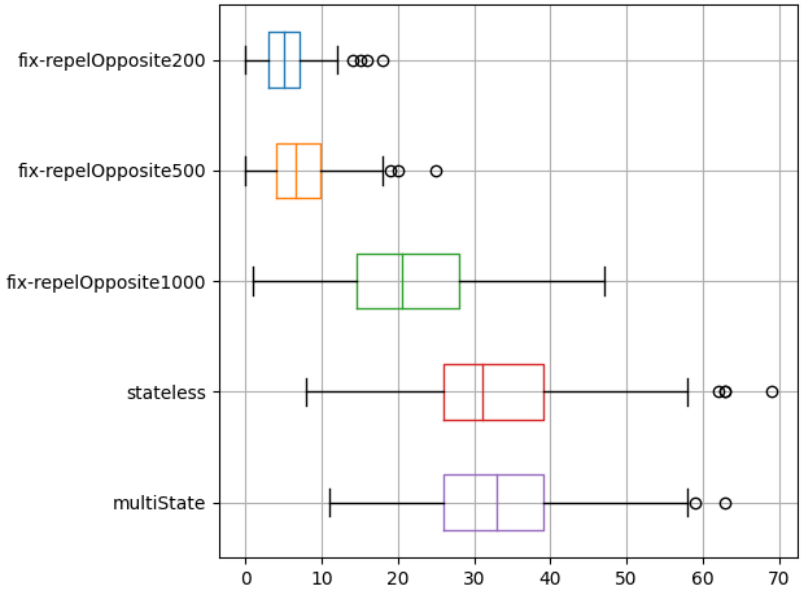

Figure 5.20: 5 Robots - Set 2 - Random Food Locations

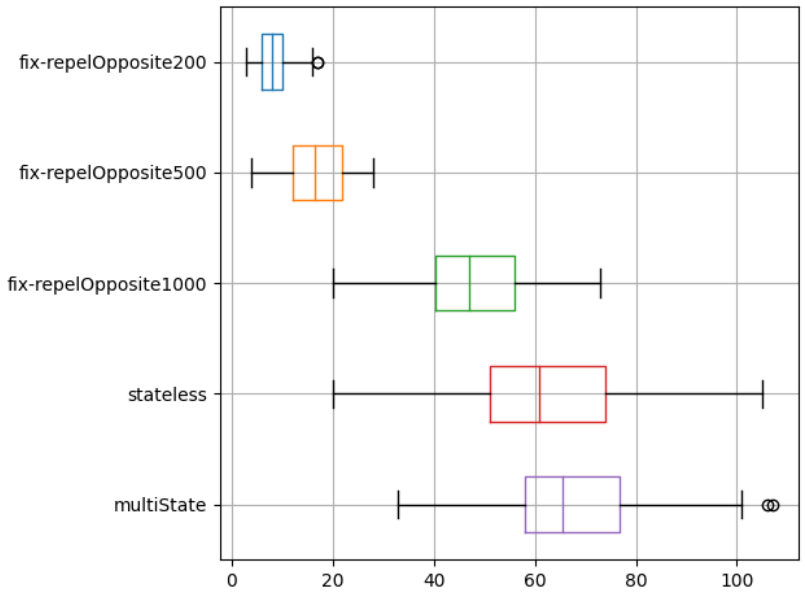

Figure 5.21: 11 Robots - Set 2 - Random Food Locations

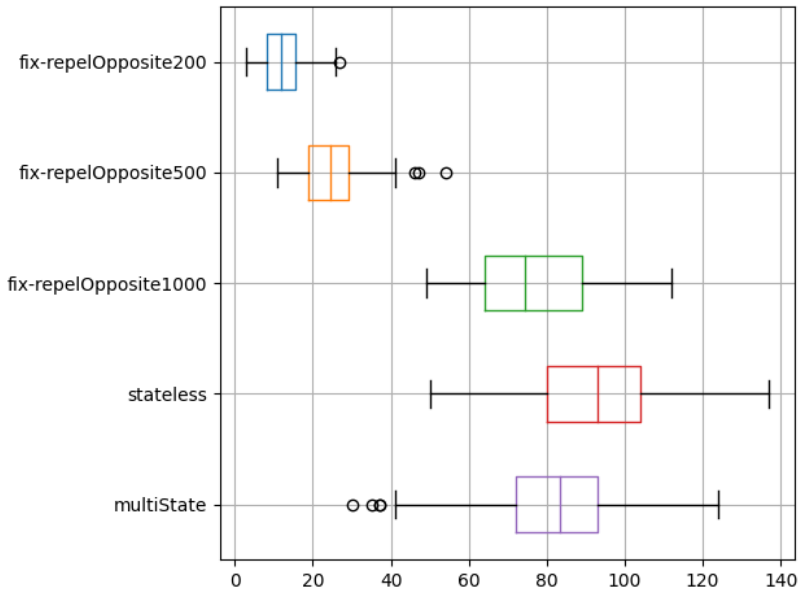

Figure 5.22: 17 Robots - Set 2 - Random Food Locations

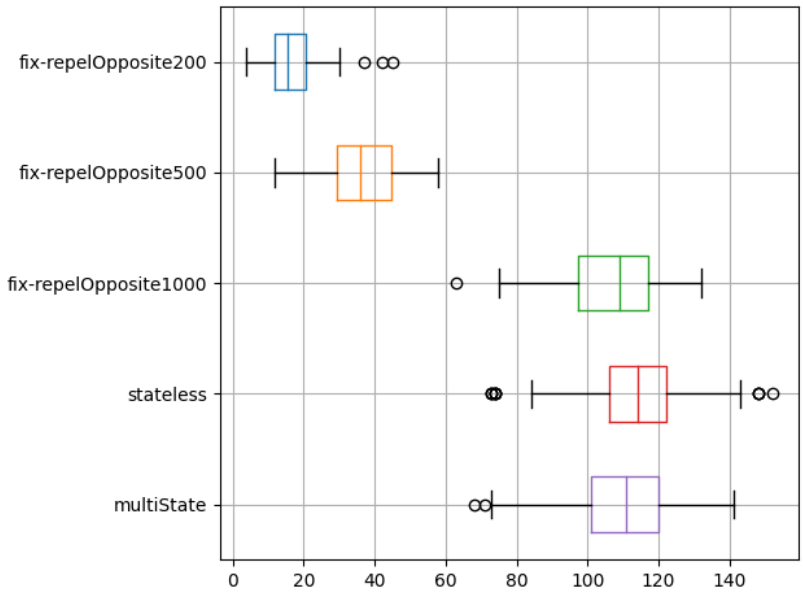

Figure 5.23: 23 Robots - Set 2 - Random Food Locations

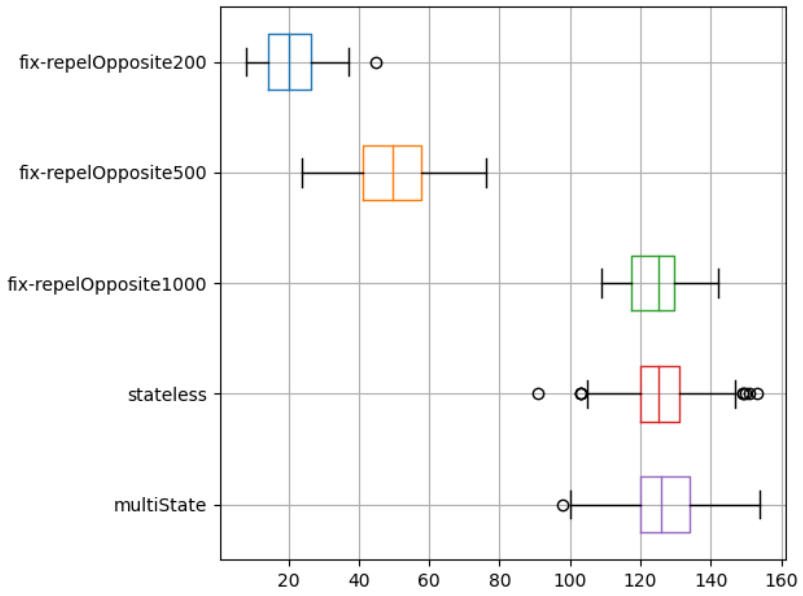

Figure 5.24: 30 Robots - Set 2 - Random Food Locations

5.3 Comparison to Previous Work

We also contrast the swarming bandits approach to previous related work. Section 5.3.1 shows the results from an empirical comparison of the swarming bandits with algorithms proposed by DWK [20]. Section 5.3.2 shows the results from a comparison with the *dUCB* (Disccounted UCB) bandit algorithm, which has been proposed to directly tackle the non-stationarity inherent in multiple robots learning in parallel (see also Chapter 6).

5.3.1 Comparison to DWK

In addressing the need to approximate knowledge of others rewards (program duration and avoidance duration), DWK tested several approximation methods for the marginal contribution of the total program time $\Delta^x P$ in Def. 4.2.1. Table 5.2 provides a brief description of each of their approximations converted to our notations. As these experiments were carried out only on one state setting, we ignore the state notation. And, as before, we use the notation a_i for individual robot action.

Figures 5.25–5.27 show the results from an empirical comparison of the DWK model [20] and its various approximation variants to our approach (marked *our approximation* in the figures). The results show that in all cases, the swarming bandits model presented here is superior to all DWK variants.

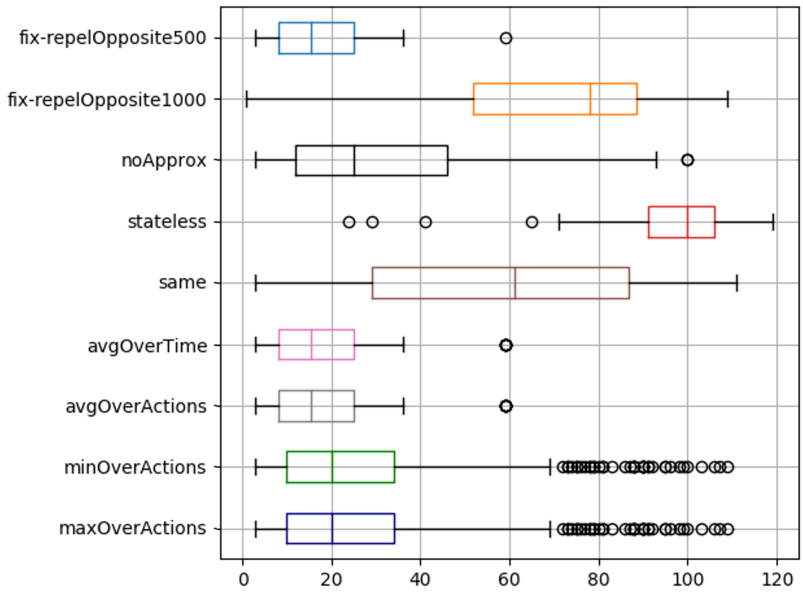

Figure 5.25: Comparisons to previous model approximations - 5 Robots - Set 1 - fixed Food Locations

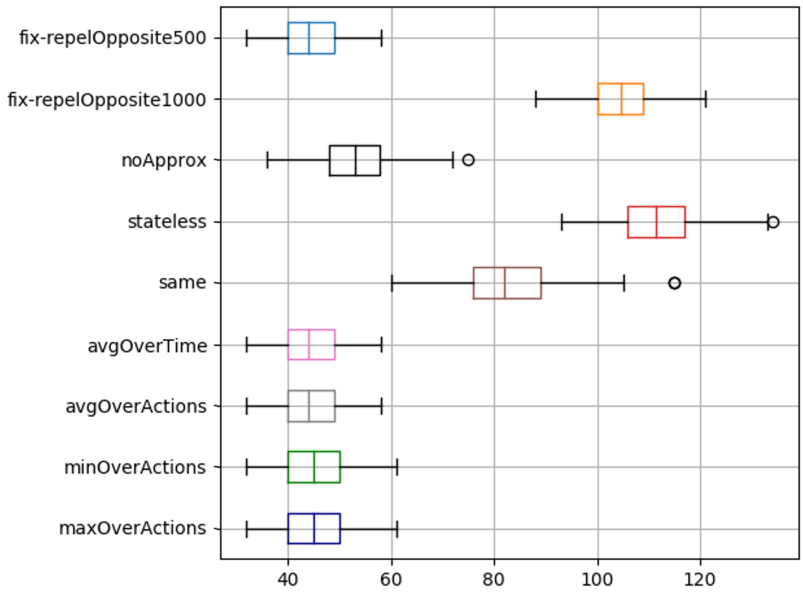

Figure 5.26: Comparisons to previous model approximations - 17 Robots - Set 1 - fixed Food Locations

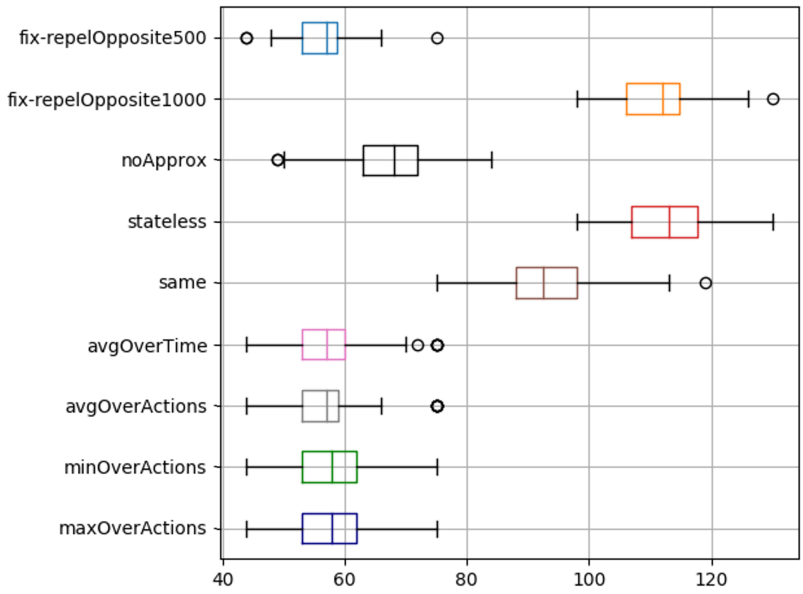

Figure 5.27: Comparisons to previous model approximations - 30 Robots - Set 1 - fixed Food Locations

Name	Description	Approx. of $\Delta^x P(a)$
Stateless	Our approximation as described in detail in Section 4.2.	$P_i(a) - n_0 \times \frac{\sum P_i(a)}{count(a)}$
No approximation	The robot ignores the others and uses only its own program time at this stage.	$P_i(a)$
Same	The robot approximates the others' program time with its own.	$P_i(a) + n_0 \times P_i(a)$
Average Over Time	The robot approximates the others' program time with its average over **all actions**.	$P_i(a) + n_0 \times \frac{\sum_{a'} \sum P_i(a')}{T}$
Average Over Actions	The robot approximates the others' program time with its average of the averages over each action.	$P_i(a) + n_0 \times \frac{1}{\lvert A_i \rvert} (\sum_{a'} \frac{\sum P_i(a')}{count(a')})$
Min Over Actions	The robot approximates the others' program time with its minimum average over each action.	$P_i(a) + n_0 \times \min_{a'} (\frac{\sum P_i(a')}{count(a')})$
Max Over Actions	The robot approximates the others' program time with its maximum average over each action.	$P_i(a) + n_0 \times \max_{a'} (\frac{\sum P_i(a')}{count(a')})$

Table 5.2: The compared approximation of the total program time.

5.3.2 Multi-Agent, Multi-Arm Bandits

The UCB algorithm which forms the basis for the swarming bandits model, assumes that rewards are stochastic—but their distribution remains stationary (fixed). This is a typical assumption in multi-arm bandit problem formulations. However, we are applying UCB in non-standard settings. In actuality, there are mulitple mulit-arm bandit problems which are present—one for each robot. Each robot selects an individual action, but its reward is a function of others' selections as well.

When multiple agents learn in parallel, there is a risk that their learning processes interfere—for instance they explore new actions independently of others, and since others do not know this, the reward they get is a function of the exploration [31]. As the agents converge to a stationary policy, the reward distribution changes as well. Thus the assumption of a stationary distribution underlying the reward is broken.

We chose to ignore this in our work, but others have proposed specific algorithms for handling non-stationary distributions. One of these is the dUCB (discounted UCB) algorithm [24], which uses a discount factor in its updating of the $Q(a)$ values, so as to minimize the effect of older values.

We compare the performance of the swarming bandit learning approach to that of dUCB, with different discounting parameters (0.5 and 0.9). Figures 5.28–5.30 show the results of these experiments. We see that the dUCB algorithms perform on par, or just below, the swarming bandit algorithm.

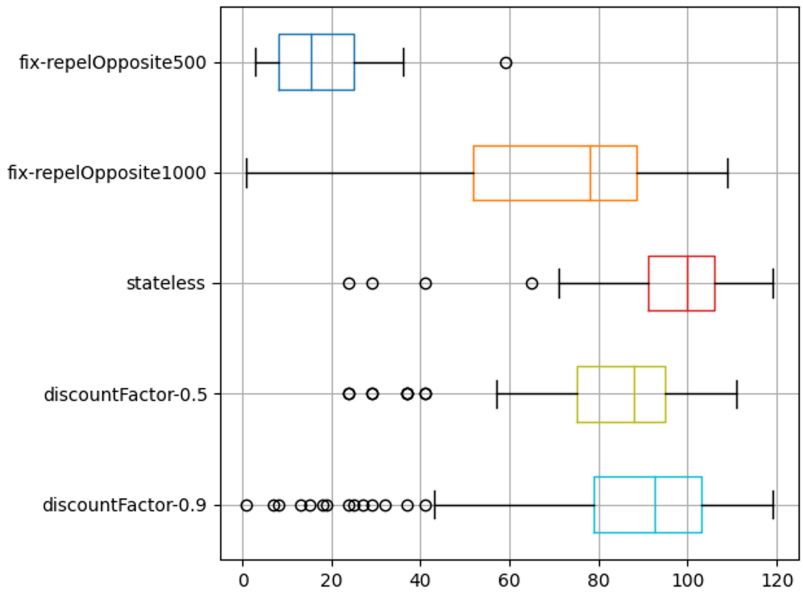

Figure 5.28: Comparisons to DUCB - 5 Robots - Set 1 - fixed Food Locations

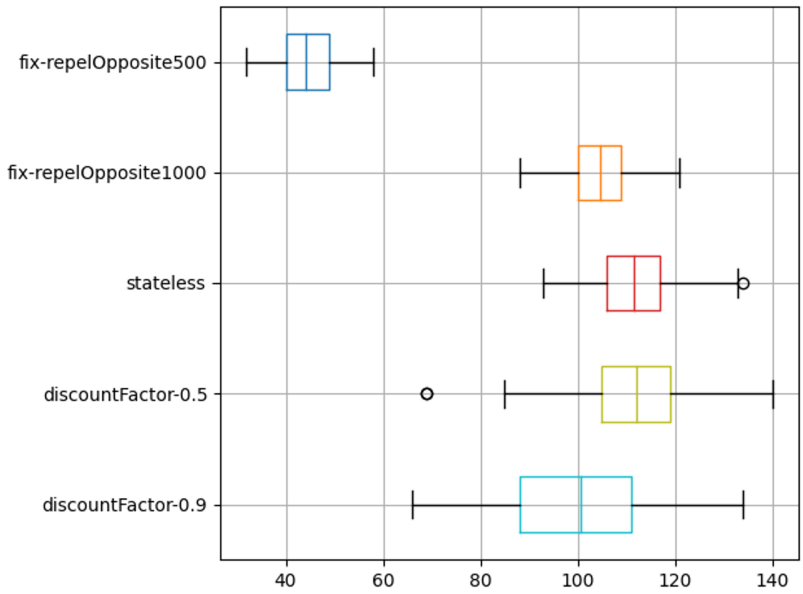

Figure 5.29: Comparisons to DUCB - 17 Robots - Set 1 - fixed Food Locations

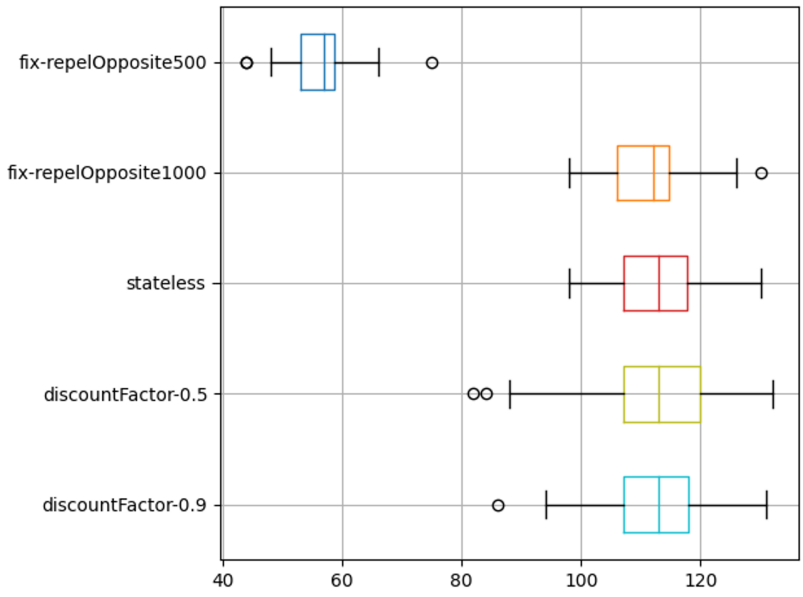

Figure 5.30: Comparisons to DUCB - 30 Robots - Set 1 - fixed Food Locations

5.4 Sensitivity to Assumptions

Several different assumptions have been made in developing the model. In this section we examine the empirical sensitivity to their violation in practice. First, we examine the assumption that utility grows with program duration (Assumption 5). Based on this assumption, we proved the same relationship exists between the total program time and the swarm achievements. But, we saw that this assumption can be easily broken and strongly depends on the quality of the implementation (Section 5.4.1). Another (implicit) assumption was that the actions the robots take to resolve conflicts only affect the coordination time. However, it appears that they may also affect the task execution (Section 5.4.2).

5.4.1 More Program time does NOT always equal higher achievements

During the experiments, we found that the behavior of the robots during the program time is very crucial to the success of the model. Accidentally, we found that it is far too easy to create task mode behaviors that break the assumption of increased productivity with increased time. For example, we found that the robots learned to walk in a circle at fixed distances from each other so they could avoid collisions entirely (see Figure 5.31). This was possible due to the fact that the behavior during the task was not good enough.

The figure shows an experiment in which red and blue LEDs were used to inform the robots when food and home bases are nearby. It was designed such that robots move around the arena while reacting to other robots and avoiding walls until they detect red (if searching for food) or blue (if searching for home) light, at which point they attempt to reach it. As light has an impact only on the area around it, the robots were able to learn how to keep distance from one another and at the same time from the food and home bases (where most collision between robots occur).

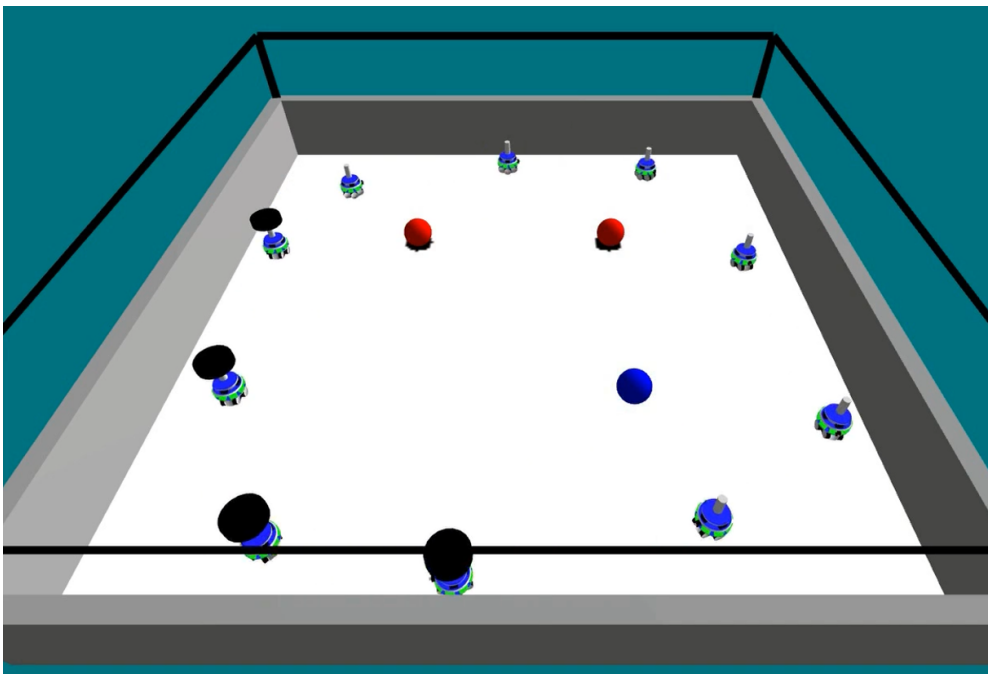

Figure 5.31: Agents learned to maximize their program time by walking in a circle

5.4.2 The Coordination Actions Also Affect The Task Execution (rather than only its time)

Another situation we saw was that the robots learn a strategy profile that leads to higher program time but lower achievements. We found out that the collision avoidance actions they could choose affected the task execution in addition to the avoidance duration. This happened, for example, when some coordination actions inadvertently point the robot away from pucks or from the home base.

Figure 5.32 illustrates the problem with an action set which includes the actions stop and repel-backward. On the left side, we see two agents colliding, where agent 1 is on its way to some goal. If agent 1 stops for two seconds, then when the two seconds are up, it will continue from the same location (right side), in constant for what happens if it repels backwards for two seconds and will be further away from the goal (middle). Note that the time division in both scenarios is the same.

Figure 5.33 shows the phenomenon more broadly, where each dot represents the average of 50 experiments where the agents react to collisions with a fix action. Three actions are compared, each with three variants, determined by the interval parameter. For example, consider the results of *fix-repelBackward1000* and *fix-BastEvade1000*, where the program time is nearly the same, but the achievements are totally different. However, as can be seen, there is a linear correlation between actions from the same type, probably because they have the same effect on the program. Due to this phenomenon, we evaluated the swarming bandits model using homogeneous action sets, where all actions are parameterized variants of the same template.

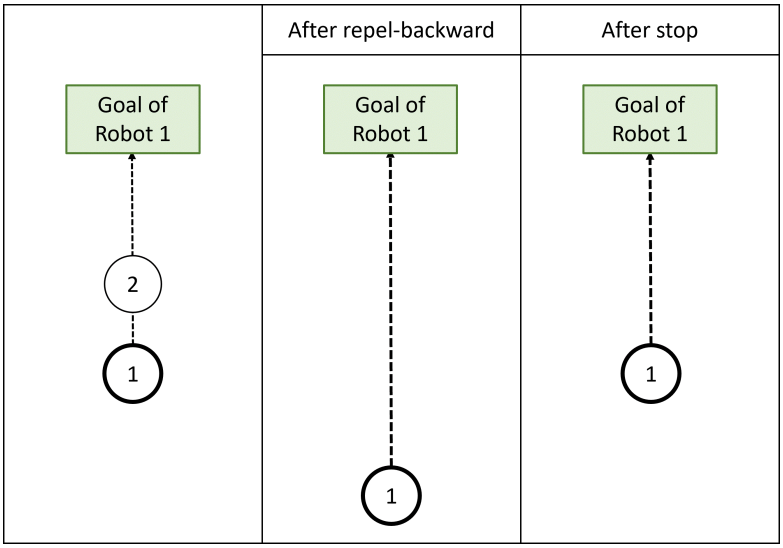

Figure 5.32: An example of a situation in which the choice affect the execution

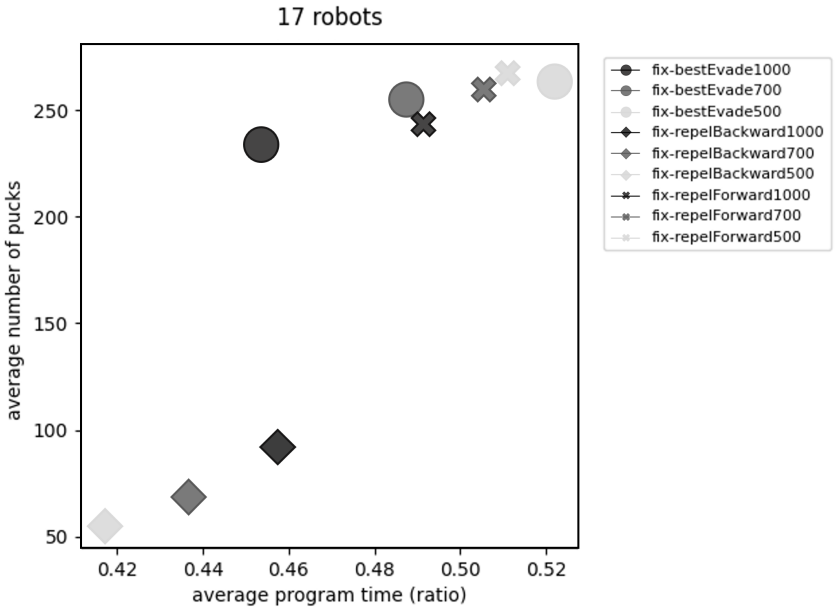

Figure 5.33: An example of different fix methods lead to different results

Chapter 6

Discussion

Several challenges are raised by the presentation of the model and the experiments: *multiple equilibria*, *non-uniform transition probabilities*, and *non-stationary* reward distribution. These are all clear challenges for future investigation. We provide some discussion of the challenges below.

Multiple Equilibria. Earlier, we showed that robots may learn a stationary strategy profile that is optimal. However, there could be several joint actions that are optimal. The model presented does not address this case.

In general, the problem of multiple equilibria [49] describes a situation where there are different optimal solutions, yet the unification of parts of them may not be an optimal one. To illustrate, assume that the best way for a swarm of two robots to cover a field is to cover its both sides simultaneously. How would they determine who would go to each side without communicating? Since they are trying to independently decide and are unaware of each other's decisions, they may go to the same direction believing that the other goes to the other direction. In game theory, this situation is called *multiple equilibria*.

Non-uniform stage-transition probabilities. We have made the assumption that the stage transition probabilities are uniformly distributed (Assumption 4). However, in practice, transition probabilities may not be distributed uniformly. When this assumption does not hold, the contextual swarming bandits may learn strategies that are optimal for each context independently, but are not optimal overall, when the expected rewards from

each stage depend also on the transition probability between one stage to the next.

We use the following simple example for intuition. Consider a game G of 100 seconds with two players, two states s_1, s_2 and two joint actions a_1, a_2. According to the state and joint action the agents spent time in coordination mode (A) and program mode (P), such that:

$$\begin{aligned} P(s_1, a_1) = 9, A(s_1, a_1) = 1, \quad P(s_1, a_2) = 8, A(s_1, a_2) = 2 \\ P(s_2, a_1) = 0, A(s_2, a_1) = 10, \quad P(s_2, a_2) = 1, A(s_2, a_2) = 9 \end{aligned} \tag{6.1}$$

The joint action a_1 always leads the agents to the state s_1 and the joint action a_2 leads to the state s_2. We get that:

$$\begin{aligned} &\forall s \in S: \quad D(s, a_1, s_1) = 1, \quad D(s, a_2, s_2) = 1 \\ &\frac{P(s_1, a_1)}{l(s_1, a_1)} = 0.9, \quad \frac{P(s_1, a_2)}{l(s_1, a_2)} = 0.8 \\ &\frac{P(s_2, a_1)}{l(s_2, a_1)} = 0, \quad \frac{P(s_2, a_2)}{l(s_2, a_2)} = 0.1 \\ &\Rightarrow \sigma^*(s_1) = a_1, \quad \sigma^*(s_2) = a_2 \end{aligned} \tag{6.2}$$

Consider comparing σ^* to a strategy profile σ' in which the agents always choose the joint action a_1. Since the transitions are fix and both strategy profiles are stationary, the course of the game depends only on the first state. If the game starts with s_1 the two plays will be the same (ten repeated stage-games from the type s_1). But if the game starts with s_2, a play that played according to σ^* yield 10 seconds of program time and a play that played according to σ' yield 81 seconds. Meaning that σ' is strictly better.

One can view this issue as some conflicts being more difficult than others, and a reasonable strategy that takes the transition probability into account might focus on avoiding them.

Non-stationary distribution of rewards: History Matters The learning algorithm we used assumes a stationary (fixed) distribution of the rewards, whereas this assumption does not necessarily hold in practice. Part of this comes from the fact that multiple robots are independently learning in parallel, something we have discussed briefly in Section 5.3.2. However, there are other causes for dynamic distribution.

For example, if the number of pucks available for foraging changes throughout the game, or pucks are farther than in late stages (e.g., because close pucks are collected first), then the distribution of rewards will not be stationary. This breaks the assumption that a stationary strategy profile can be optimal, as the history of the stages matters.

Chapter 7

Conclusions

The thesis presented a game-theoretic model of cooperative swarms, and proscribed a practical approach—-*swarming bandits*—for swarm robots to optimize their joint goal. The model is intended for swarm activities in which interactions—where coordination is needed—interfere with the swarm goals. It models activities in which swarm goals are maximized when the swarm members are given as much opportunity as possible to act without interference.

Under these conditions, it is possible to describe the swarm as engaging in infinite-horizon stochastic games, where the goal is to maximize the time spent on *direct work*, i.e., the total time spent by the swarm members on the task, without the overhead of handling collisions and interactions with others. A swarm that behaves according to the model is guaranteed to achieve the maximal utility, when agents act rationally. The use of time opens the possibility for individuals to assess progress, as they can measure their own time spent on the task and in handling collisions.

We showed how the stochastic game model, which is descriptive, can be used in a prescriptive form to guide the practical individual decision-making. Rationally maximizing the individual reward derived from the model optimizes the swarm-level goals.

The model can be used in practice under some conditions and practical approximations, but it also relies on parameters which are never known in advance. To overcome these unknowns, we use multi-arm bandits as a reinforcement-learning framework that guides online selection of coordination actions. We modified the classic UCB algorithm for continuous time, and demonstrated that it works well in extensive experiments, carried out in

a sophisticated 3D physics-based simulation often used for swarm robotics research. Finally, we discussed open challenges that remain for future work.

Appendix A

The LocustLib - Open Source Library For Krembot-Sim

LocustLib was designed for Krembot-Sim and provides an easy way for the user to work with our model as well as to adapt and extend it. In the Krembot-Sim, Krembot controllers are simulating. Each of them have a loop function that is activated at each step and determines what it will do based on the information that it receives from its sensors. As was described in detail in the previous chapters, our model is divided into two parts - program and coordination, and respectively the controller holds two separate components, one for each. The class diagram is shown in figure A.1 and the sequence diagram is shown in figure A.2.

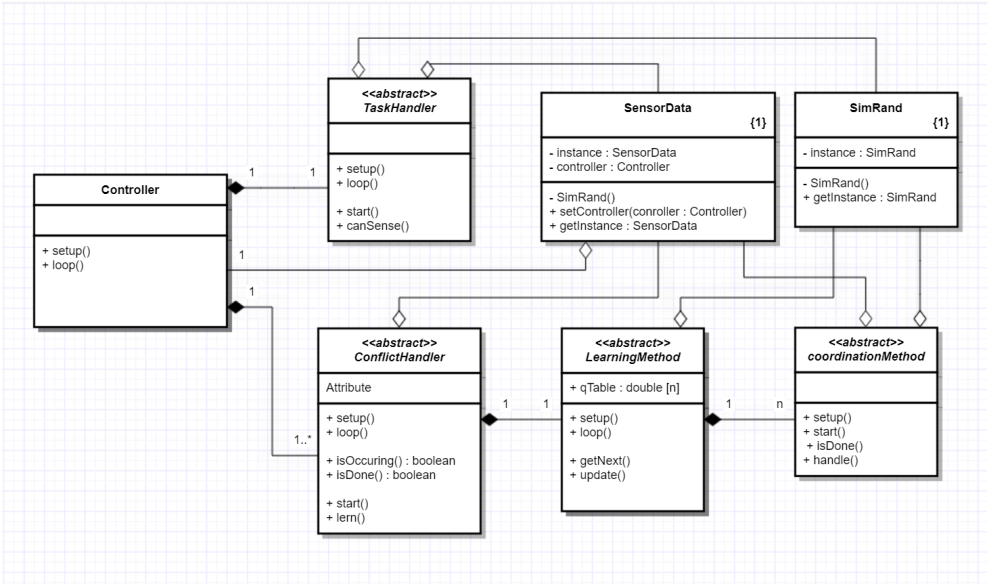

Figure A.1: LocustLib - class diagram

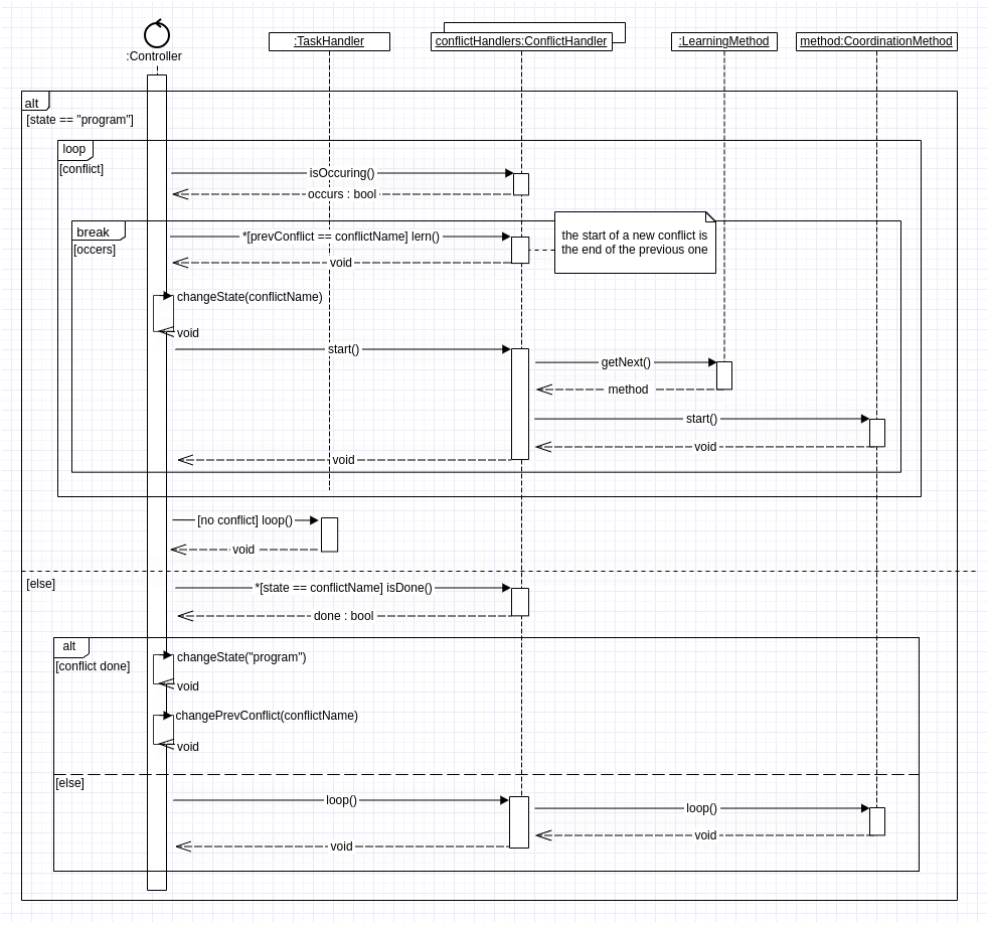

Figure A.2: Sequence diagram - controller main loop

Bibliography

[1] M. Agarwal, V. Aggarwal, and K. Azizzadenesheli. Multi-Agent Multi-Armed Bandits with Limited Communication. *arXiv:2102.08462 [cs]*, Feb. 2021. arXiv: 2102.08462.

[2] N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for autonomous mobile robots. *SIAM Journal on Computing*, 36(1):56–82, 2006.

[3] M. Aitken, G. Butler, D. Lemmon, E. Saindon, D. Peters, and G. Williams. The Lord of the Rings: the visual effects that brought middle earth to the screen. In *ACM SIGGRAPH 2004 Course Notes*, SIGGRAPH '04, pages 11–es, New York, NY, USA, Aug. 2004. Association for Computing Machinery.

[4] I. Aoki. A Simulation Study on the Schooling Mechanism in Fish. *Nippon Suisan Gakkaishi*, 48(8):1081–1088, 1982.

[5] G. Ariel and A. Ayali. Locust Collective Motion and Its Modeling. *PLOS Computational Biology*, 11(12):e1004522, Oct. 2015. Publisher: Public Library of Science.

[6] G. Ariel, Y. Ophir, S. Levi, E. Ben-Jacob, and A. Ayali. Individual Pause-and-Go Motion Is Instrumental to the Formation and Maintenance of Swarms of Marching Locust Nymphs. *PLOS ONE*, 9(7):e101636, Feb. 2014. Publisher: Public Library of Science.

[7] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2):235–256, May 2002.

[8] L. Bayındır. A review of swarm robotics tasks. *Neurocomputing*, 172:292–321, Jan. 2016.

[9] J.-P. Benoit and V. Krishna. Finitely Repeated Games. *Econometrica*, 53(4):905–22, 1985. Publisher: Econometric Society.

[10] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, Apr. 2002.

[11] D. S. Calovi, U. Lopez, S. Ngo, C. Sire, H. Chaté, and G. Theraulaz. Swarming, schooling, milling: phase diagram of a data-driven fish school model. *New Journal of Physics*, 16(1):015026, Jan. 2014. Publisher: IOP Publishing.

[12] M. Chakraborty, K. Y. P. Chua, S. Das, and B. Juba. Coordinated Versus Decentralized Exploration In Multi-Agent Multi-Armed Bandits. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 164–170, Melbourne, Australia, Aug. 2017. International Joint Conferences on Artificial Intelligence Organization.

[13] M. Chen, F. Dai, H. Wang, and L. Lei. DFM: A Distributed Flocking Model for UAV Swarm Networks. *IEEE Access*, 6:69141–69150, 2018. Conference Name: IEEE Access.

[14] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98/IAAI '98, pages 746–752, USA, July 1998. American Association for Artificial Intelligence.

[15] V. Conitzer and T. Sandholm. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67(1-2):23–43, May 2007.

[16] A. Czirók, A.-L. Barabási, and T. Vicsek. Collective Motion of Self-Propelled Particles: Kinetic Phase Transition in One Dimension. *Physical Review Letters*, 82(1):209–212, Jan. 1999. Publisher: American Physical Society.

[17] C. Delgado-Mata, J. I. Martinez, S. Bee, R. Ruiz-Rodarte, and R. Aylett. On the Use of Virtual Animals with Artificial Fear in Virtual Environments. *New Generation Computing*, 25(2):145–169, Feb. 2007.

[18] M. Dorigo, M. Birattari, and M. Brambilla. Swarm robotics. *Scholarpedia*, 9(1):1463, 2014. revision #138643.

[19] Y. Douchan and G. A. Kaminka. The effectiveness index intrinsic reward for coordinating service robots. In S. Berman, M. Gauci, E. Frazzoli, A. Kolling, R. Gross, A. Martinoli, and F. Matsuno, editors, *13th International Symposium on Distributed Autonomous Robotic Systems (DARS-2016)*. Springer, November 2016.

[20] Y. Douchan, R. Wolf, and G. A. Kaminka. Swarms can be rational. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2019.

[21] N. El Houda Bahloul, S. Boudjit, M. Abdennebi, and D. E. Boubiche. A Flocking-Based on Demand Routing Protocol for Unmanned Aerial Vehicles. *Journal of Computer Science and Technology*, 33(2):263–276, Mar. 2018.

[22] M. Fontan and M. Matarić. Territorial multi-robot task division. *IEEE Transactions of Robotics and Automation*, 14(5):815–822, 1998.

[23] N. Fridman and G. Kaminka. Modeling pedestrian crowd behavior based on a cognitive model of social comparison theory. *Computational & Mathematical Organization Theory*, 16:348–372, Dec. 2010.

[24] A. Garivier and E. Moulines. On Upper-Confidence Bound Policies for Non-Stationary Bandit Problems. *arXiv:0805.3415 [math, stat]*, May 2008. arXiv: 0805.3415.

[25] A. Garivier and E. Moulines. On upper-confidence bound policies for switching bandit problems. In *International Conference on Algorithmic Learning Theory*, pages 174–188. Springer, 2011.

[26] M. Gauci, R. Nagpal, and M. Rubenstein. Programmable Self-disassembly for Shape Formation in Large-Scale Robot Collectives. In R. Groß, A. Kolling, S. Berman, E. Frazzoli, A. Martinoli, F. Matsuno,

and M. Gauci, editors, *Distributed Autonomous Robotic Systems*, volume 6, pages 573–586. Springer International Publishing, Cham, 2018. Series Title: Springer Proceedings in Advanced Robotics.

[27] K. Genter, N. Agmon, and P. Stone. Ad hoc teamwork for leading a flock. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 531–538. Citeseer, 2013.

[28] D. Goldberg and M. Matarić. Interference as a tool for designing and evaluating multi-robot controllers. In *AAAI/IAAI*, pages 637–642, 1997.

[29] E. T. Hall. A System for the Notation of Proxemic Behavior1. *American Anthropologist*, 65(5):1003–1026, 1963. _eprint: https://anthrosource.onlinelibrary.wiley.com/doi/pdf/10.1525/aa.1963.65.5.02a00020.

[30] C. Hartman and B. Benes. Autonomous boids: Research Articles. *Computer Animation and Virtual Worlds*, 17:199–206, July 2006.

[31] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote. A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity. *arXiv:1707.09183 [cs]*, Mar. 2019. arXiv: 1707.09183.

[32] G. A. Kaminka, D. Erusalimchik, and S. Kraus. Adaptive multi-robot coordination: A game-theoretic perspective. In *2010 IEEE International Conference on Robotics and Automation*, pages 328–334, May 2010. ISSN: 1050-4729.

[33] G. A. Kaminka and N. Fridman. Simulating urban pedestrian crowds of different cultures. *ACM Transactions on Intelligent Systems and Technology*, 9(3):27:1–27:27, 2018.

[34] G. A. Kaminka, R. Spokoini-Stern, Y. Amir, N. Agmon, and I. Bachelet. Molecular robots obeying Asimov's three laws of robotics. *Artificial Life*, 23(3):343–350, 2017.

[35] T. T. J. Kiran. Prompt Review of State of the Art of Swarm Intelligence Framework and Use Cases. *International Journal of Application or Innovation in Engineering & Management*, 9(10):20, 2020.

[36] J. Langford and T. Zhang. The Epoch-Greedy Algorithm for Multi-armed Bandits with Side Information. In J. Platt, D. Koller, Y. Singer,

and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[37] M. L. Littman. Value-function reinforcement learning in Markov games. *Cognitive Systems Research*, 2(1):55–66, Apr. 2001.

[38] U. Lopez, J. Gautrais, I. D. Couzin, and G. Theraulaz. From behavioural analyses to models of collective motion in fish schools. *Interface Focus*, 2(6):693–707, Dec. 2012. Publisher: Royal Society.

[39] U. Madhushani and N. E. Leonard. A Dynamic Observation Strategy for Multi-agent Multi-armed Bandit Problem. In *2020 European Control Conference (ECC)*, pages 1677–1682, Saint Petersburg, Russia, May 2020. IEEE.

[40] M. J. Matarić. Reinforcement Learning in the Multi-Robot Domain. *Autonomous Robots*, 4(1):73–83, Mar. 1997.

[41] L. Matignon, G. J. Laurent, and N. Le Fort-Piat. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1):1–31, Feb. 2012.

[42] J. F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, Jan. 1950. Publisher: National Academy of Sciences Section: PNAS Classic Article.

[43] M. J. Osborne. *An introduction to game theory.* Oxford University Press, New York, 2004. OCLC: 51769105.

[44] E. Ostergaard, G. Sukhatme, and M. Matarić. Emergent bucket brigading. In *Autonomous Agents*, 2001.

[45] D. W. Oyler, Y. Yildiz, A. R. Girard, N. I. Li, and I. V. Kolmanovsky. A game theoretical model of traffic with multiple interacting drivers for use in autonomous vehicle development. In *2016 American Control Conference (ACC)*, pages 1705–1710, July 2016. ISSN: 2378-5861.

[46] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo. ARGoS: a Modular, Parallel,

Multi-Engine Simulator for Multi-Robot Systems. *Swarm Intelligence*, 6:271–295, Dec. 2012.

[47] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4):25–34, Aug. 1987.

[48] P. Romanczuk, I. D. Couzin, and L. Schimansky-Geier. Collective Motion due to Individual Escape and Pursuit Response. *Physical Review Letters*, 102(1):010602, Jan. 2009. Publisher: American Physical Society.

[49] G. Romp. *Game Theory: Introduction and Applications*. Oxford University Press, 1997.

[50] A. Rosenfeld, G. A. Kaminka, S. Kraus, and O. Shehory. A study of mechanisms for improving robotic group performance. *Artificial Intelligence*, 172(6–7):633–655, 2008.

[51] A. Rosenfeld, G. A. Kaminka, S. Kraus, and O. Shehory. A study of mechanisms for improving robotic group performance. *Artificial Intelligence*, 172(6):633–655, Apr. 2008.

[52] M. Rubenstein, C. Ahler, and R. Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *2012 IEEE International Conference on Robotics and Automation*, pages 3293–3298, St Paul, MN, USA, May 2012. IEEE.

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

[54] S. Shahrampour, A. Rakhlin, and A. Jadbabaie. Multi-armed bandits in multi-agent networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2786–2790, New Orleans, LA, Mar. 2017. IEEE.

[55] L. S. Shapley. Stochastic Games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, Oct. 1953. Publisher: National Academy of Sciences Section: Mathematics.

[56] A. Shiloni, N. Agmon, and G. A. Kaminka. Of robot ants and elephants: A computational comparison. *Theoretical Computer Science*, 412(41):5771–5788, Sept. 2011.

[57] A. Slivkins. Introduction to Multi-Armed Bandits. *arXiv:1904.07272 [cs, stat]*, June 2021. arXiv: 1904.07272.

[58] E. Solan. Acceptable strategy profiles in stochastic games. *Games and Economic Behavior*, 108:523–540, 2018.

[59] Y. Sugiyama, M. Fukui, M. Kikuchi, K. Hasebe, A. Nakayama, K. Nishinari, S.-i. Tadaki, and S. Yukawa. Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam. *New Journal of Physics*, 10(3):033001, Mar. 2008. Publisher: IOP Publishing.

[60] Y. Tan and Z.-y. Zheng. Research Advance in Swarm Robotics. *Defence Technology*, 9(1):18–39, Mar. 2013.

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

[62] T. Vicsek, A. Czirok, E. Ben-Jacob, I. Cohen, and O. Sochet. Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75(6):1226–1229, Aug. 1995. arXiv: cond-mat/0611743.

[63] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. E. Eiben, and T. Vicsek. Optimized flocking of autonomous drones in confined environments. *Science Robotics*, 3(20), July 2018. Publisher: Science Robotics Section: Research Article.

[64] W. H. Warren. Collective Motion in Human Crowds. *Current Directions in Psychological Science*, 27(4):232–240, Aug. 2018. Publisher: SAGE Publications Inc.

[65] D. H. Wolpert and K. Tumer. An Introduction to Collective Intelligence. *arXiv:cs/9908014*, Aug. 1999. arXiv: cs/9908014.

[66] C. Zhang and V. Lesser. Coordinating Multi-Agent Reinforcement Learning with Limited Communication. In *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1101–1108, St. Paul, MN, 2013. IFAAMAS.

[67] B. Zhou, X. Tang, and X. Wang. Learning Collective Crowd Behaviors with Dynamic Pedestrian-Agents. *International Journal of Computer Vision*, 111(1):50–68, Jan. 2015.

[68] M. Zuluaga and R. Vaughan. Reducing spatial interference in robot teams by local-investment aggression. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Alberta, August 2005.

[69] E. Şahin. Swarm Robotics: From Sources of Inspiration to Domains of Application. In E. Şahin and W. M. Spears, editors, *Swarm Robotics*, pages 10–20, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

Hebrew Abstract