Master Thesis

Competitive Multi-Swarm Systems

Author:
Karen Katz

Supervisor:
Prof. Gal A. Kaminka

Department of Computer Science
Bar-Ilan University

March 25, 2023

Acknowledgments

I would like to express my sincere gratitude to my advisor, Prof. Gal A. Kaminka for his outstanding support and guidance throughout my master's thesis journey. Prof. Kaminka's expertise, encouragement, and mentorship were invaluable to me as I learned crucial research skills, as well as critical thinking and academic writing skills. His support has helped me grow both personally and academically, and I am grateful for his dedication to my success.

In addition, I would like to express my appreciation to the members of the MAVERICK Research Group for their valuable feedback and support during this time period. I am particularly grateful to Dr. Hana Weitman for her support and assistance throughout my time in the group and while working on this thesis.

I would also like to thank the Department of Computer Science faculty and staff for providing me with the resources needed to complete my academic degree. Special thanks to Ms. Dafna Gad, for helping me with administrative processes throughout my academic studies.

Last but not least, I would like to extend a special thanks to my family for their support throughout the way. Your faith in me has been a source of motivation, promoting my growth and development in every aspect of my life.

Abstract

A competitive multi-swarm system, is a system with two or more distinct swarms of simple agents, with limited local knowledge, sharing the same environment and resources, where each swarm's goal is to outperform the other swarms.

In this work, we developed a general model for multiple competitive swarms from game theoretic perspective. We formulated the individual and global utilities for K-swarms competition, based only on a single assumption of zero-sum game between two individual players from different swarms. A special case of two swarms competition is shown to be a zero-sum game, and a possible extension to zero-sum game for the K-swarm case is presented.

To show the applicability of the model, the theory is applied into the field of competitive robot swarms. Global and individual utilities, and the estimation of the individual player's impact on its surroundings are presented as a function of times, to support applicability to any generic task. One result from this approach is that a robot can increase its swarm's utility not only by performing its original task, but also by interfering in its opponents' performance of their tasks.

We propose a learning process for each individual robot in multi-swarm competition, by calculating its own reward, and providing a general way for evaluation and selection of its possible actions. The proposed learning model tries to overcome the gap due to the partial information known to each robot, by considering the swarm identity of the other robots during each interaction, and approximating differences between the swarms.

As an example, the general model is applied for the more specific sub-field of multi-swarm competitive foraging. It examines the model on the unexplored problem of how robots in a competitive multi-swarm environment should interact during spatial conflicts, in order to outperform the other swarms.

The proposed model has been validated and tested through an extensive series of simulated experiments, including two- and three-swarm competitions, in various densities, with and without learning. Part of the experiments were expanded for cases of a learning swarm with initial disadvantages. The results show that a learning swarm performed at least equally and usually better than a non-learning swarm, which uses a predefined policy. Surprisingly, in many cases, the overall score of all the swarms together increased when competition was involved.

Table of Contents

List of Figures

List of Tables

Nomenclature

Symbol	Meaning
$\mathcal{S} = \{S_1, ...S_K\}$	Set of players in game of K swarms
$N = \sum N_i$	Total number of players
$a_i \in \mathcal{A}$	A specific action a taken by player i in an interaction, from set $\mathcal{A}$ of possible actions
$p_i \in \mathcal{P}$	A specific program action p executed by player i, between two interactions, from set $\mathcal{P}$ of possible actions
$s_i \in \mathcal{S} = \{1, .., K\}$	The state (collision type) of player i in an interaction
$a = (a_1, ...a_N)$	An action profile: the joint action composed of the players' actions in a specific interaction
u_i^m	The utility of player i from interacting with a player from swarm S_m
$U_i = \sum_{m=1}^{K} u_i^m$	The accumulated utility of a player i, obtained by all of its interactions with all K swarms
$U_{S_k} = \sum_{i \in S_k} U_i$	The accumulated utilities obtained by all N_k members of swarm S_k
$\mathbb{U}_{S_k} = U_{S_k} - \sum_{\substack{m=1 \\ m \neq k}}^{K} U_{S_m}$	The *Swarm Utility Difference (SUD)* of swarm S_k
$T(a_i)$	Action time of player i in an interaction
$T(p_i)$	Program time of player i in an interaction
$T \in \mathbb{R}$	Total game time for all players
$\pi_i \in \mathcal{A}^K$	The policy of actions player i uses in all K types of interactions
$\pi_{S_k} : \pi_1 \times ... \times \pi_{N_k} \in \mathcal{A}^{K \times N_k}$	The policy of swarm S_k

Table 2: List of symbols and notations used in this thesis.

Chapter 1

Introduction

Swarms are collectives of multiple simple agents with only limited and local communications, and no central control [29, 53, 7]. The agents typically act towards a common goal and share resources. Swarms are easily found in nature, for example in bird flocks and fish schools. Such natural behaviors inspire artificial systems, e.g., in *Swarm Robotics* [46, 57, 6, 5].

A *Competitive Multiple Swarm* system, is a system with two or more distinct swarms, sharing the same environment and resources. In such systems, each swarm's goal is to score better than the other swarms, where score is quantified in some task performance measure. Unfortunately, there is little research on multiple competitive swarms (see Chapter 2 for more details).

We take *competitive foraging* as an example for the task to be performed by the competing swarms. *Foraging* is a domain of swarm robotics, where robots are repeatedly searching and collecting items, and carrying them to a home base [30, 62, 34]. Foraging results are usually measured by counting the number of collected items, by the whole swarm. In *competitive* swarm foraging, the winning swarm is the one that collectively brought the highest number of items to their home base.

In this work we develop a game-theoretic model for multiple competitive swarms. We concentrate on robots' spatial interactions, and more specifically on the challenge of how robots in a competitive multi-swarm environment should interact during spatial conflicts, to outperform the other swarms.

This is challenging because the robots cannot communicate with each other. Thus, global information such as the collective score of a swarm or the total number of members in each swarm, is unknown to any of its member agents.

We apply reinforcement learning methods to train robots individually in a competitive multi-swarm environment. The proposed learning model tries to overcome the challenges due to the partial information known to each robot, by considering the swarm identity of the other robots during each interaction.

This work is organized as follows. Chapter 2 provides background and review of related work. It first reviews swarm robotics, cooperation and competition in swarm systems, and multiple robot coordination. It then reviews reinforcement learning in the context of multi-agent systems. The chapter concludes by reformulating previously proposed competitive coordination model notation, to fit the context of this work.

Chapter 3 takes a game-theoretic view for the case of multiple swarms. It starts with extensive form game which is then represented as a series of normal-form game, using zero-sum game assumption for competitive games. The chapter then formulates the global utilities, concluded by finding the individual player's contribution to the swarm utility difference, while taking into account also the impact of the absence of a player.

Chapter 4 applies the theory introduced in the previous chapter to competitive robot swarms. It defines the global and individual utilities as a function of time for a generic multi-swarm competition, by estimating the individual player's utility and the effect on its surroundings. It then defines the learning process of each individual robot in multi-swarm competition, by calculating its own reward, and providing a general way for evaluation and selection of its possible actions.

Chapter 5 uses the proposed model for a more specific case of multi-swarm competitive foraging, and proposes an experimental environment and settings. It adopts the previously proposed model to competitive foraging, and defines relevant action sets and policies. Concludes by proposing approximations to the reward functions, due to partial information of the individual robot.

Chapters 6 and 7 describe and discusses experiments and results which support the proposed theory and learning model. Chapter 6 examines various baseline cases using the proposed models and validates the assumptions made as a basis for the model. Chapter 7 elaborates on the case of non-identical program executing behavior for different swarms, and in particular when one of the swarm has some kind of advantage over the other, and a case of three swarms competition.

Finally, Chapter 8 discusses gaps found in this work, open questions and thoughts raised during experiments, and propose directions for future work. Chapter 9 concludes and summarizes this work.

Chapter 2

Background and Related Work

Our work is related to several areas in multi-robot and multi-agent systems. We review related work in these areas, and their relation to this work. We review swarm robotics, cooperation and competition in swarm systems, and multiple robot coordination. We then review reinforcement learning in the context of multi-agent systems, and conclude by reformulating previously proposed competitive coordination model notation, to fit the context of this work.

2.1 Swarm Robotics

Swarm robotics is an approach to robotics that takes inspiration from the self-organized behaviors of social animals and insects, which provide fascinating examples of how a large number of simple individuals can interact to create collectively intelligent systems [46]. Through simple rules and local interactions, swarm robotics seeks to generate robust, scalable, and coherent collective behaviors for large numbers of robots [8]. In contrast with traditional *multi-robot systems* which use centralized or hierarchical control and communication systems, that allow all robots in the system to coordinate with each other, swarm robotics adopts a decentralized approach in which the desired collective behaviors emerge from the local interactions between robots, and between robots and their environment [47]. Therefore, in a swarm-robotic system the robots' sensing and communication capabilities are local, and do not have access to global knowledge. Essentially, single-swarm of robots addresses cooperative behavior, acting towards a mutual goal, even though the individual robots may not be aware of it. Robot swarms can either be *homogeneous* or *heterogeneous* in terms of hardware, software, behavior, parameters and goals [28].

Examples of swarm tasks include coverage [45, 50], patrolling [19, 31], collective motion [60, 69], mapping [44, 16] and more. Many practical and potential applications of swarm robotics are unmanned aerial vehicles (UAVs) [12], spacecrafts [37], autonomous underwater vehicles (AUVs) [49], ground mobile robots [3], and other swarm-robotic based applications in hazardous [40] or unknown [24] environments.

A canonical task of swarm robotics is *Foraging* [34, 67, 13, 55]. As described in [62], foraging robots are mobile robots capable of searching for, and when found transporting objects to, one or more collection points, called home bases. Foraging may be carried out by an individual (i.e., a single robot in the swarm), or in groups (multi-robot swarm), as in our work. Examples of potential real-world applications of foraging robots include mine-clearing, hazardous waste clean-up, and search and rescue tasks [51]. In this work, we use foraging as a basis for a multi-swarm competition.

A major challenge in swarm-robotic systems, is resolution of spatial interactions or conflicts. Spatial conflicts must be resolved since robots cannot share the same spot at the same time. There are many approaches to collision avoidance and resolution, each may influence differently on the overall performance. There is no single perfect policy for collision avoidance in swarms [43]. Therefore a reinforcement learning based approach is used to adjust the reactive coordination method to use in each conflict [18, 27]. In this work we propose a model for the open question of how robots should act during spatial interactions in settings of multiple competing swarms to best perform.

It is also challenging to estimate the influence of a specific action in a limited information system. The work by Erusalimchik et al. [27] has examined the use of the ratio of collision avoidance time as a substitute for the robot's estimation of its own utility. In particular, minimizing this ratio was proposed to be an alternative to maximizing the swarm utility. However, this conjectured connection was not satisfactorily proven in their work. The work by Rosenfeld et al. [43], shows that there is a strong correlation between interaction costs and group performance.

The more a robot, or a group of robots invest on the global task, the lower their conflicts duration, and therefore, their performance is higher. Thus, the gains of the swarm, and of each robot individually, are proportional to the total program execution time of the swarm. We build on Douchan et al. [18] and assume that the utility of a player from an interaction is a linear combination of both the conflict and the program execution durations. We expand the latter to the competitive multi-swarm case in Chapter (4).

2.2 Cooperation and Competition in Swarms Systems

Collective behaviors of complex systems emerge from local interactions among individuals [25]. The interaction type can be either cooperative or competitive. In this work we investigate the case of at least two distinct swarms, and the influence of interaction type on the overall performance.

Cooperative behavior refers to interaction among robots along with increasing the system's overall utility. Hence, all the robots in the system interact and work for a common goal or reward [59]. Various illustrative examples of cooperation within a single swarm, are motion planning [17], foraging [30], and cooperative search [38]. Another example of a single-swarm system is shown in [42], questioning whether cooperative or competitive behavior between agents with a common goal should be preferred, resulting with a preference for cooperation. We preserve the cooperation within the swarm, and expand the problem to a multi-swarm system case. In this work we demonstrate that a certain type of a competitive behavior between distinct swarms is either equal or better than cooperative behavior both for the competitive swarm, and for the overall performance of all swarms.

Competitive behavior, which is the opposite of cooperative behavior, refers to the case in which multiple robots compete among themselves in order to satisfy their own individual interest [59]. Examples of competition between individual robots are [20, 48]. Another different field related to competitive behavior is multiple agents in adversarial environments. The works [1, 66] discuss area coverage in a known adversarial environment, or patrolling along a perimeter to increase an intruder detection probability. However, these works did not consider swarms.

In a competitive swarm behavior, each swarm's goal is to outperform the other swarms. Some works dealing with multiple swarms competition, are [15], where two UAV swarms are trying to destroy each other; In [14], one swarm defends an area by collectively surrounding the attackers' swarm; and [52], where the swarm of defenders try to block the other swarm from intruding into a protected area. The last three competitive swarm works are different from ours as follows: the first two do not include game theoretic analysis and approach, and the last one does not involve learning.

We address multi-swarm competitive settings. All swarms have the same goal, which is to compete on the same limited resources and outperform any other swarm. Previous works [18, 27, 43, 22] have been applied to cooperative single swarm systems where the goal is to achieve the highest possible score. However, in competition, a better score may be achieved by interfering with the other team, preventing them from achieving scores, by concentrating on scoring, or using a combination of both. In multiple swarm systems, spatial interactions occur either within the swarm (intra-swarm collision) or between swarms (inter-swarm collision). Therefore, techniques and models from the above works, cannot directly be applied to multi-swarm competitive systems.

2.3 Reinforcement Learning in context of Multi-Agent Systems

An agent using reinforcement learning (**RL**) learns by interacting with its dynamic environment [10, 61, 26, 54]. At each time step, the agent perceives the state of the environment and takes an action, which causes the environment to transit into a new state. A scalar reward signal evaluates the quality of each transition, and the agent has to maximize the cumulative reward along the course of interaction [10].

Multi-agent (**MA**) learning has received attention in the past years [68, 23, 56, 9, 65, 32, 39]. When designing multi agent systems, it is impossible to predict all the potential situations agents may encounter and specify all agents' behaviors optimally in advance. Therefore, agents in such systems should learn from, and adapt to their operating environment and their counterparts [64].

Multi-agent reinforcement learning (**MARL**) may be used either due to the complexity or the decentralization of the system. [9, 39, 64]. We use such techniques in this work, where agents may learn either to coordinate or compete with other learners.

The game-theoretic analysis and experiments of single-swarm systems in [18, 27, 43] show convergence to significantly improved results using reinforcement learning, when compared with a predetermined fixed baseline. These works showed that multiple action policies within a single swarm can be beneficial to the overall score of the swarm. We hypothesized and show in this work that the improvement using MARL resulting with multiple action policies, can be achieved in competitive multi-swarms systems. However, the difference between the single cooperative swarm of the previous work, vs. competitive multi-swarm systems in this work, requires distinguishing and treating differently inter-swarm and intra-swarm interactions. Distinguishing also influences rewards, evaluations, policies and behaviors.

The work of Yang and Wang, [65] shows learning in zero-sum games, but for individuals rather than for swarms. An approach to reinforcement learning in multi-agent general-sum games is described in [33], where a learner is told to treat each other agent as either a "friend" or "foe". Unlike our work, this work analyzed the case for multiple individual agents, maximizing their individual reward, and has only two types of players. We were inspired by this approach during this research, and expand to swarms of players, where the considered reward is aggregated over the swarm rather than individual player. Furthermore, we expand the approach from two types to K types, i.e., K different swarms.

An approach we use in this work is the *Wonderful Life Utility* (**WLU**), presented in [63]. WLU is the marginal contribution made by the agent to the global utility. In other words, it is the difference between the group utility with the agent, and without it. It is known that agents that learn with WLU as a utility function play a potential game with the global utility as the potential function [35, 2]. WLU is expected to make each agent's utility more learnable by removing the unnecessary dependencies on other agents' assignment decisions, while keeping the agent's utilities aligned with the global utility.

To conclude, there are multiple previous works on each of the related subfields. However, the challenge of how robots should act during spatial interactions in settings of competitive multi-swarm system, is still open.

Chapter 3

A Game Theoretic View of Multiple Competitive Swarms

A *Competitive Multiple Swarm System*, is a system with two or more distinct swarms, sharing the same environment and resources, where each swarm's goal is to outperform any of the other swarms. Each swarm is composed of agents with limited knowledge, and the swarm score is function of all of its members' individual scores.

This chapter applies a game-theoretic view to competitive multiple swarm systems. In Section 3.1 we present the game between competitive swarms as an extensive-form zero-sum game. Under certain assumptions this is transformed into a sequence of normal-form games of different types, i.e., a Markov game. We then define the different types of games based on the swarms involved, and further formulate the utilities of individual agents and of each swarm, in Section 3.2. In Section 3.3 we use the Wonderful Life Utility (**WLU**) function to determine the marginal contribution of an individual player to the utility, to overcome the limited knowledge of an individual.

3.1 Game Models of Competitive Swarms

A swarm is composed of agents with limited knowledge, and with local awareness of their surroundings: the agents do not have any global information about the game, and cannot communicate with any other agent. Agents interact with other agents, resulting in individual utilities. The aggregation of all individual utilities is of the swarm's global utility. The swarms seek to each achieve greater global utility than the others.

Previous works [27, 18] modeled a single swarm cooperative task as *a game*, which interactions occur between *players*. Each single player selects an action to be performed from a predefined *set*. Individual utilities are generated from the combination of the individual actions.

We expand the representation of the single cooperative swarm to the case of multiple competitive swarms, as a basis for this work. In a multiple swarm competitive game, it is important to distinguish between interactions with players of the same swarm (*intra-swarm* interactions), to cooperate with, and players from a different swarm (*inter-swarm* interactions), to compete with. In this work we assume that a player can identify the swarm identity of other players with which it interacts, but not their individual identity.

In the general case, there are K competing swarms. We denote the K participating swarms as $\mathcal{S} = \{S_1, ..., S_K\}$, where each swarm $S_{\mathrm{k}} \in \mathcal{S}$, is comprised of N_k players, with total number of players from all swarms, of $N = \sum_k N_k$. In each interaction between players, each player selects an action from a predefined set, $a \in \mathcal{A}$. Once an interaction has been concluded, each of the participant players, is rewarded an individual utility value, marked u. The global utility of a swarm, is a function of all individual utilities of its players.

We use a simple running example of a game with two competing swarms, to illustrate. Fig. 3.1 shows the extensive form game, for two competing swarms, assuming interactions of two types, either intra- or inter-swarm, marked *Friend* or *Foe,* respectively, and a set of two possible actions to perform in each interaction, marked as *Left* or *Right.* Here we assume that there are at least three players in the game, the player of interest, P_i, and at least another player from each swarm, marked P_j from other swarm, and P_l from the swarm of player P_i.

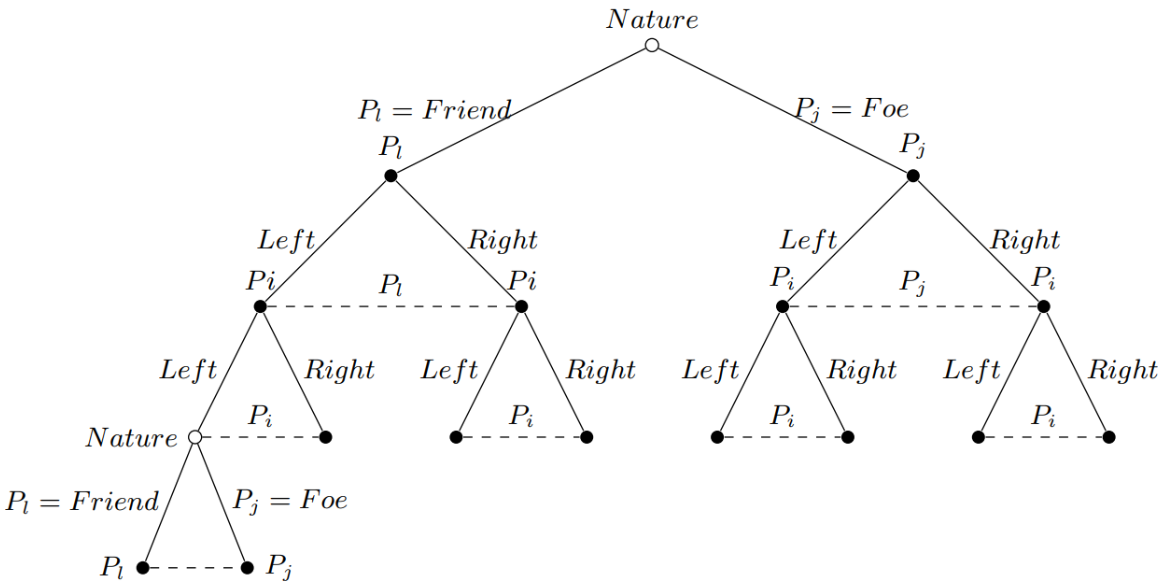


Figure 3.1: Extensive form game of a player interacting with a friend or a foe, for the case of two competing swarms.

In this example, the root node of the game tree, represents the interaction type, known to the players, marked as a selection of *Nature*, i.e., none of the participants can select the identity of the opponent player. The next two layers represent the simultaneous choices of player P_i and the other player, P_j or P_l, for action. The players are aware of the other players' swarm-identities, but not their choices of actions. The process described above in the first three layers, repeats for any interaction until the game ends. Note that at the end of each interaction, all participants are rewarded with a utility value, which contributes to their swarm utility, as detailed in the next subsection.

In principle, the utilities of the players can depend on the history of the states and joint actions played from the game start until the current interaction. We follow previous work [18] in assuming the Markov property, meaning that given a game state, the utilities do not depend on the complete history of all joint actions performed, but rather on the game state immediately preceding the interaction, and the actions taken by the players in the current interaction. This also means that no matter what interaction it is, as long as the state and the joint actions remain the same, the outcome remains the same. Under this assumption, we can transform the extensive form game into a series of normal form games.

We mark with k the swarm identity of the individual player P_i, i.e., $i \in S_k$. Similarly, we mark the other player, P_j, and its swarm as m, i.e., $j \in S_m$. We mark the two possible actions as 'L' and 'R', and mark a_i, a_j as the actions performed by players P_i, P_j respectively. We also mark u_i and u_j as the utilities of the players from the current interaction.

Table 3.1 presents a two-player game (interaction), with two possible actions for each player. Under the Markov property, for competitive swarms, the swarm identity of a player's opponent, affects the interaction state. Therefore, each individual utility in Table 3.1, depends on the other player's swarm identity, as well as both actions performed.

$i \in S_k, j \in S_m$	$a_j = L$	$a_j = R$
$a_i = L$	$u_i(m,(L,L)), u_j(k,(L,L))$	$u_i(m,(L,R)), u_j(k,(L,R))$
$a_i = R$	$u_i(m,(R,L)), u_j(k,(R,L))$	$u_i(m,(R,R)), u_j(k,(R,R))$

Table 3.1: Folded Game Matrix for two players, multiple swarms and two possible actions assuming the Markov property.

For each swarm, there are K different versions of Table 3.1, since there are K different swarms in the competitive swarm game model. Every matrix represents a possible interaction type, depending on the swarm identity of the opponent player. In our example, since $K = 2$, we therefore need two matrices, one for inter-swarm interaction and one for intra-swarm interaction, where the identities are known to the participating players.

For intra-swarm interactions, where the two interacting players belong to the same swarm, i.e., $i, j \in S_k$, Table (3.2a) shows the utility outcome of a cooperative game where the swarm accumulates utility over time.

For the inter-swarm interactions, where $i \in S_k, j \in S_m, m \neq k$, it is logical to assume that as a result of interaction between opponent players of competing swarms, one player's gain is the other's loss, i.e., a zero-sum game. Table (3.2b) shows the utility outcome of a competitive game. Note the zero-sum of the individual utilities in each cell. We use this assumption for the rest of this work.

$i, j \in S_k$	$a_j = L$	$a_j = R$
$a_i = L$	$u_i((L,L)), u_j((L,L))$	$u_i((L,R)), u_j((L,R))$
$a_i = R$	$u_i((R,L)), u_j((R,L))$	$u_i((R,R)), u_j((R,R))$

(a) Utility of a two-player cooperative intra-swarm game presented as a General-sum game.

$i \in S_k, j \in S_m, m \neq k$	$a_j = L$	$a_j = R$
$a_i = L$	$u_i((L,L)), -u_i((L,L))$	$u_i((L,R)), -u_i((L,R))$
$a_i = R$	$u_i((R,L)), -u_i((R,L))$	$u_i((R,R)), -u_i((R,R))$

(b) Utility of a two-player competitive inter-swarm game presented as a Zero-sum game.

Table 3.2: Example of utility matrices of two actions for inter- and intra-swarm interactions.

We now formulate the utility for a single player and for a swarm based on the assumptions demonstrated in Tables (3.2a) and (3.2b). Namely, for an *inter-swarm* interaction, an interaction between two players, each belongs to a different swarm, we assume a zero-sum game, meaning that for such interaction, the gain of one player is the loss of the others. For an *intra-swarm* interaction, an interaction between two players from the same swarm, we assume a general-sum game as in previous works.

For a single inter-swarm interaction, between player $i \in S_k$, and player $j \in S_m$, where $k \neq m$, we denote the utility obtained by player i from interacting with player from swarm S_m, and the utility obtained by player j from interacting with player from swarm S_k, as u_i^m and u_j^k, respectively. Similarly, for intra-swarm interactions, between players of the same swarm, we denote the utilities as u_i^k and u_j^m, for the case where player i interacted with some other player from the same swarm, S_k, and player j interacted with a player from its own swarm, S_m.

Definition 1. Based on the zero-sum game assumption, for a single inter-swarm interaction (Table 3.2b), we can rewrite the value of each player's utility:

$$u_i^m = -u_j^k \tag{3.1}$$

From Eq. (3.1) above, for every swarm pair, (S_k, S_m), where $k \neq m$, the accumulated utility of all S_k players from interactions with players from S_m is the negative accumulated utility of all S_m players from interactions with players from S_k:

$$\sum_{i \in S_k} u_i^m = -\sum_{j \in S_m} u_j^k \tag{3.2}$$

To generalize for the case of K competitive swarms in a game, one may use one of the \$$K$\$-player zero-sum game methods of *Polymatrix Game* or *Stochastic Multi-player Game*, described in [58, 11].

To summarize, we have expanded the single swarm cooperative model into a competitive K-swarm game, and transformed the extensive form game into a sequence of K-types repeated games, i.e., a Markov game[1]. We also distinguished between cooperative intra-swarm interaction, within each of the swarms, which can be described as a general-sum game, and assumed a zero-sum game for competitive inter-swarm interactions.

3.2 Formulating Global Utilities

We now analyze the utilities gained by the swarms, and their relation to the individual players' utilities.

Definition 2. We denote U_i, as the accumulated utility of a single player $i \in S_k$, obtained by all of its interactions with all K swarms:

$$U_i = \sum_{k=1}^{K} u_i^k \tag{3.3}$$

We can split the accumulated utility of each player to the sum of inter-swarm and intra-swarm utilities:

$$U_i = \sum_{k=1}^{K} u_i^k = u_i^k + \sum_{\substack{m=1 \\ m \neq k}}^{K} u_i^m \tag{3.4}$$

[1] We assume that any of the K game types are equally likely.

Definition 3. The accumulated utilities obtained by all N_k members of swarm S_k, is denoted as U_{S_k}:

$$U_{S_k} = \sum_{i \in S_k} U_i \tag{3.5}$$

We can split the total utility of each swarm to the sum of inter-swarm and intra-swarm utilities, using Eq. (3.3), (3.4) and (3.5):

$$U_{S_k} = \sum_{i \in S_k} U_i = \sum_{i \in S_k} \sum_{k=1}^{K} u_i^k = \sum_{i \in S_k} u_i^k + \sum_{i \in S_k} \sum_{\substack{m=1 \\ m \neq k}}^{K} u_i^m \tag{3.6}$$

The performance of a swarm in a competitive multi-swarm system, should relate to the performance of the other swarms in the game. We define the *Swarm Utility Difference (**SUD**)* function to evaluate the swarms' performances.

Definition 4. We define the *Swarm Utility Difference (**SUD**)* of swarm S_k, marked as $\mathbb{U}_{S_k}$, as the difference between the accumulated utilities of all players in S_k, and the accumulated utilities of all other players in all other swarms $S_{m \neq k}$.

$$\mathbb{U}_{S_k} = U_{S_k} - \sum_{\substack{m=1 \\ m \neq k}}^{K} U_{S_m} \tag{3.7}$$

In the general multiple swarm case, where there are K swarms, and no constraints on the total utility gained by any swarm, U_{S_k}, we show that when using $\mathbb{U}_{S_k}$, the game is not necessarily a zero-sum game. This results from the general-sum property of intra-swarm interactions.

Theorem 5. *The game between $K > 2$ swarms is not necessarily a zero-sum game.*

Proof. By summing all *Swarm-Utilities* using Eq. (4) we can write:

$$\begin{aligned}
\sum_{k=1}^{K} \mathbb{U}_{S_k} &= \sum_{k=1}^{K} \left(U_{S_k} - \sum_{m=1, m \neq k}^{K} U_{S_m} \right) \\
&= \sum_{k=1}^{K} U_{S_k} - \sum_{k=1}^{K} \left(\sum_{m=1}^{K} U_{S_m} - U_{S_k} \right) \\
&= \sum_{k=1}^{K} U_{S_k} - \sum_{m=1}^{K} \sum_{k=1}^{K} U_{S_k} + \sum_{k=1}^{K} U_{S_k} \\
&= 2 \cdot \sum_{k=1}^{K} U_{S_k} - K \cdot \sum_{k=1}^{K} U_{S_k} \\
&= (2 - K) \cdot \sum_{k=1}^{K} U_{S_k} = (2 - K) \cdot U \qquad (3.8)
\end{aligned}$$

where U is the sum of all swarm's utilities: $U = \sum_{k=1}^{K} U_{S_k}$. □

The sums of all Swarm-utilities in a K-swarm game depends on the sum of the swarms' total utilities, U_{S_k}, and the number of playing swarms, K. We see that Theorem (6) is a special case of

We now claim and show that for the two swarm case ($K = 2$), the whole game becomes a zero-sum game, using the SUD function.

Corollary 6. *In the case of $\mathcal{S} = \{S_k, S_m\}$, the game between the two swarms with utility function of $\mathbb{U}_{S_k}$ is a zero-sum game.*

Proof. Using definition (4), we can write:

$$\begin{aligned}
\mathbb{U}_{S_k} &= U_{S_k} - U_{S_m} \\
\mathbb{U}_{S_m} &= U_{S_m} - U_{S_k}
\end{aligned}$$

Therefore, the sum of the Swarm-utilities is zero:

$$\mathbb{U}_{S_k} + \mathbb{U}_{S_m} = (U_{S_k} - U_{S_m}) + (U_{S_k} - U_{S_m}) = 0$$

□

Theorem (6) shows that for the general case of unconstrained game, where $K = 2$, it is guaranteed regardless of any specific individual, or accumulated utility functions, that the game between two swarm is a zero-sum game.

We have shown in Theorem 5 that unlike the two-swarm case, in the general case of $K > 2$ competitive swarms, the game is not necessarily a zero-sum game. However, we now show that under the following assumption, this work can still be applicable to $K > 2$ swarms, and even show experimental results for such case, later in this work.

Under the assumption of constant-sum game, e.g., where the total resources are limited and the end of the game is reached when all resources has been used, the following theorem shows how to modify such a K-swarm game into a zero-sum game.

Theorem 7. *Under the assumption of a limited global-utility in a game, the game between K swarms is a constant-sum game and therefore can be modified to a K+1 swarms zero-sum game.*

Proof. Assigning the assumption that the accumulated utilities obtained by all players in all swarms is constant, *C,* we can write:

$$U = \sum_{k=1}^{K} U_{S_k} = C$$

Using Eq. (3.8), we can further write:

$$\begin{aligned}\sum_{k=1}^{K} \mathbb{U}_{S_k} &= (2-K) \cdot \sum_{k=1}^{K} U_{S_k} \\ &= (2-K) \cdot C\end{aligned}$$

By adding a new swarm S_{K+1} with a constant total-utility of $U_{S_{K+1}} = -C$, by using Eq. (3.7) we get:

$$\begin{aligned}
\sum_{k=1}^{K+1} \mathbb{U}_{S_k} &= \sum_{k=1}^{K+1} \left(U_{S_k} - \sum_{m=1,m\neq k}^{K+1} U_{S_m} \right) \\
&= \sum_{k=1}^{K+1} U_{S_k} - \sum_{k=1}^{K+1} \left(\sum_{m=1}^{K+1} U_{S_m} - U_{S_k} \right) \\
&= \sum_{k=1}^{K+1} U_{S_k} - \sum_{k=1}^{K+1} \sum_{m=1}^{K+1} U_{S_m} + \sum_{k=1}^{K+1} U_{S_k} \\
&= 2 \cdot \sum_{k=1}^{K+1} U_{S_k} - (K+1) \cdot \sum_{m=1}^{K+1} U_{S_m} \\
&= (1-K) \cdot \sum_{k=1}^{K+1} U_{S_k} \\
&= (1-K) \cdot \left(\sum_{k=1}^{K} U_{S_k} + U_{S_{K+1}} \right) \\
&= (1-K) \cdot (C - C) = 0
\end{aligned}$$

Therefore, the game of the swarms set $\mathcal{S} = \{S_1, S_2, ..., S_K\}$, where the global-utility is constant, $\sum_{k=1}^{k=K} U_{S_k} = C$, can be converted to a zero-sum game by adding a swarm S_{k+1} with a constant Total-utility, $U_{S_{K+1}} = -C$. □

To summarize the subsection, we defined different swarm utility functions based on the individual players' utilities. We have shown that any competitive two-swarm game is a zero-sum game, and generalized for the case with more than two swarms.

3.3 Finding Individual's Contribution to the Swarms' Utilities

In the previous subsection, we formulated the utility of individual players and of a swarm. However, players are neither aware of the other players' actions nor of their utilities. As a result they cannot choose their individual actions to improve their Swarm Utility Difference. To overcome this, we propose to transform the game into a potential game.

In a Potential Game, the incentive of all players to change their strategy can be expressed using a single global function called the potential function [36]. A Potential Game is a normal form game where for every player i, the difference in the utility of every unilateral deviation of player i's action a_i is related to the difference of a single potential function $\psi(a)$ mapping joint actions to a scalar. Potential games can solve the challenge of converging to the optimal action while requiring only data from each player.

In the context of multi-agent coordination and to facilitate learning, we need to find a reward function for each player based on its intrinsic measurements in a way that the agent plays a potential game. This will make the players converge to an optimal joint action.

The *Wonderful Life Utility* (**WLU**), presented in [63] is the marginal contribution made by the agent to the global utility. In other words, it is the difference between the group utility with the agent, and without it. It is known that agents that use WLU as a utility function play a potential game with the global utility as the potential function [35, 2]. WLU is expected to make each agent's utility aligned without needing to consider dependencies on other agents' decisions, while keeping the agent's utilities aligned with the global utility.

Definition 8. The WLU of player $\rho \in S_k$, using the SUD defined in Eq. (4), is:

$$WLU_{\rho}^{\mathbb{U}_{S_k}} = W_{\rho}^{\mathbb{U}_{S_k}} = \mathbb{U}_{S_k} - \mathbb{U}_{S_k/\{\rho\}} \tag{3.9}$$

In the rest of this subsection, we find the Wonderful Life Utility of player $\rho \in S_k$, $W_{\rho}^{\mathbb{U}_{S_k}}$ of Eq. (3.9). The contribution of player ρ to $\mathbb{U}_{S_k}$, which is the difference between the swarm utilities, results from its contribution to both its own utility, and the other swarms' utilities. In a previous subsection we calculated $\mathbb{U}_{S_k}$. We now explore $\mathbb{U}_{S_k/\{\rho\}}$, the SUD of S_k, where player ρ is absent.
We define the effect of player ρ on the utility of swarm S_k, denoted as $\phi_{S_k}^{\rho}$.

Definition 9. The effect of player ρ on swarm S_k, which is the marginal contribution of ρ to S_k, is:

$$\phi^{\rho}_{S_k} = \epsilon^{\rho}_{S_k} - \epsilon^{\tilde{\rho}}_{S_k} \tag{3.10}$$

where $\epsilon^{\rho}_{S_k}$ is the utility got by the swarm when ρ is present, and $\epsilon^{\tilde{\rho}}_{S_k}$ is the alternative utility that would have gotten by the swarm when ρ is absent.

Lemma 10. *$\epsilon^{\rho}_{S_m} = -u^m_{\rho}$, when $\rho \in S_k$, $k \neq m$.*

Proof. Due to the zero-sum game assumption of Eq. (3.1) and Eq. (3.11), and summing over all of the interactions between player ρ ans players from swarm $S_{m \neq k}$. □

In other words, the utilities achieved by swarm S_m due to the presence of player $\rho \in S_k$, is the negative of the sum of the utilities got by player ρ due to the presence of swarm S_m.

To provide some intuition about swarm utilities, using the zero-sum game assumption, we explore the case of two swarms competition. This restriction may assist in evaluating the individual's contribution to the swarms' utilities.

Corollary 11. *For the case of two swarms $\mathcal{S} = \{S_k, S_m\}$, ($K = 2$), the SUD of swarm S_k is:*

$$\mathbb{U}_{S_k} = \sum_{i \in S_k} u^k_i - \sum_{j \in S_m} u^m_j + 2 \cdot \sum_{i \in S_k} u^m_i \tag{3.11}$$

Proof. For the case of two swarms, we can rewrite Eq. (3.6) and (3.7) as:

$$\mathbb{U}_{S_k} = U_{S_k} - U_{S_m} \tag{3.12}$$

$$U_{S_k} = \sum_{i \in S_k} \sum_{k=1}^{2} u^k_i = \sum_{i \in S_k} u^k_i + \sum_{i \in S_k} u^m_i \tag{3.13}$$

$$U_{S_m} = \sum_{j \in S_m} \sum_{m=1}^{2} u^m_j = \sum_{j \in S_m} u^m_j + \sum_{j \in S_m} u^k_j \tag{3.14}$$

Therefore, assigning Eq. (3.2) into Eq. (3.14):

$$U_{S_m} = \sum_{j \in S_m} u_j^m - \sum_{i \in S_k} u_i^m \tag{3.15}$$

Assigning equations (3.15) and (3.13) into Eq. (3.12)

$$\begin{aligned}\mathbb{U}_{S_k} = U_{S_k} - U_{S_m} &= \sum_{i \in S_k} u_i^k + \sum_{i \in S_k} u_i^m - \left(\sum_{j \in S_m} u_j^m - \sum_{i \in S_k} u_i^m \right) \\ &= \sum_{i \in S_k} u_i^k - \sum_{j \in S_m} u_j^m + 2 \cdot \sum_{i \in S_k} u_i^m\end{aligned}$$

□

From Corollary (11), we observe that the Swarm Utility Difference (SUD) of swarm S_k equals to the sum of the intra-swarm utilities of all players from S_k, minus the the sum of the intra-swarm utilities of all players from S_m, plus twice the inter-swarm utilities of all players of S_k.

In order to evaluate the contribution of a single player to its swarm, we first evaluate the SUD in the absence of that player.

Corollary 12. *For the case of two swarms, the SUD of swarm S_k when $\rho \in S_k$ is absence is:*

$$\mathbb{U}_{S_k/\{\rho\}} = \sum_{i \in S_k} u_i^k - \sum_{j \in S_m} u_j^m + 2 \cdot \sum_{i \in S_k/\{\rho\}} u_i^m - \phi_{S_k}^{\rho}$$

Proof. By definition, the total utility of swarm S_k, is the sum of: (1) the utilities achieved by $S_k/\{\rho\}$ when ρ is absent, (2) the utility achieved by ρ, and (3) the effect of ρ on players $S_k/\{\rho\}$.

$$U_{S_k} = U_{S_k/\{\rho\}} + U_\rho + \phi_{S_k}^{\rho}$$

Therefore,

$$U_{S_k/\{\rho\}} = U_{S_k} - U_\rho - \phi_{S_k}^{\rho} \tag{3.16}$$

Assigning equations (3.6) and 3.4 into Eq. (3.16):

$$\begin{aligned}
U_{S_k/\{\rho\}} &= \left(\sum_{i\in S_k} u_i^k + \sum_{i\in S_k} u_i^m\right) - \left(u_\rho^k + u_\rho^m\right) - \phi_{S_k}^\rho \\
&= \sum_{i\in S_k} u_i^k + \sum_{i\in S_k} u_i^m - u_\rho^k - u_\rho^m - \phi_{S_1}^\rho \\
&= \sum_{i\in S_k/\{\rho\}} u_i^k + \sum_{i\in S_k/\{\rho\}} u_i^m - \phi_{S_1}^\rho \qquad (3.17)
\end{aligned}$$

Similarly, the total utility of swarm S_m, is the sum of: (1) the utilities achieved by S_m when $\rho \in S_k$ is absent, and (2) the utilities achieved by S_m when colliding with ρ:

$$U_{S_m} = U_{S_m/\{\rho\}} + \phi_{S_m}^\rho = U_{S_m/\{\rho\}} + \epsilon_{S_m}^\rho - \epsilon_{S_m}^{\tilde{\rho}}$$

Therefore,

$$U_{S_m/\{\rho\}} = U_{S_m} - \epsilon_{S_m}^\rho + \epsilon_{S_m}^{\tilde{\rho}} \qquad (3.18)$$

Therefore, using Eq. (3.2) and Lemma (10),

$$\begin{aligned}
U_{S_m/\{\rho\}} &= \sum_{j\in S_m} u_j^m + \sum_{j\in S_m} u_j^k - \epsilon_{S_m}^\rho + \epsilon_{S_m}^{\tilde{\rho}} \\
&= \sum_{j\in S_m} u_j^m + \sum_{i\in S_k} (-u_i^m) - \left(-u_\rho^m\right) + \epsilon_{S_m}^{\tilde{\rho}} \\
&= \sum_{j\in S_m} u_j^m - \sum_{i\in S_k} u_i^m + u_\rho^m + \epsilon_{S_m}^{\tilde{\rho}} \\
&= \sum_{j\in S_m} u_j^m - \sum_{i\in S_k/\{\rho\}} u_i^m + \epsilon_{S_m}^{\tilde{\rho}} \qquad (3.19)
\end{aligned}$$

Assigning (3.17) and (3.19) into Corollary (11),

$$\begin{aligned}
\mathbb{U}_{S_k/\{\rho\}} &= U_{S_k/\{\rho\}} - U_{S_m/\{\rho\}} \\
&= \sum_{i\in S_k/\{\rho\}} u_i^k - \phi_{S_k}^\rho + \sum_{i\in S_k/\{\rho\}} u_i^m
\end{aligned}$$

$$
\begin{aligned}
&- \left(\sum_{j \in S_m} u_j^m - \sum_{i \in S_k/\{\rho\}} u_i^m + \epsilon_{S_m}^{\tilde{\rho}} \right) \\
&= \sum_{i \in S_k/\{\rho\}} u_i^k - \sum_{j \in S_m} u_j^m + 2 \cdot \sum_{i \in S_k/\{\rho\}} u_i^m - \phi_{S_k}^{\rho} - \epsilon_{S_m}^{\tilde{\rho}}
\end{aligned}
$$

□

Our final step is evaluating WLU function of a single player $\rho \in S_k$, based on the above.

Theorem 13. *In the case of two swarms, where* $\rho \in S_k$:

$$W_\rho^{\mathbb{U}_{S_k}} = u_\rho^k + 2 \cdot u_\rho^m + \phi_{S_k}^{\rho} + \epsilon_{S_m}^{\tilde{\rho}}$$

Proof. Assigning Corollaries (11) and (12) into Definition (8), we can write:

$$
\begin{aligned}
W_\rho^{\mathbb{U}_{S_k}} &= \left(\sum_{i \in S_k} u_i^k - \sum_{j \in S_m} u_j^m + 2 \cdot \sum_{i \in S_k} u_i^m \right) \\
&\quad - \left(\sum_{i \in S_k/\{\rho\}} u_i^k - \sum_{j \in S_m} u_j^m + 2 \cdot \sum_{i \in S_k/\{\rho\}} u_i^m - \phi_{S_k}^{\rho} - \epsilon_{S_m}^{\tilde{\rho}} \right) \\
&= \left(\sum_{i \in S_k} u_i^k - \sum_{i \in S_k/\{\rho\}} u_i^k \right) + 2 \cdot \left(\sum_{i \in S_k} u_i^m - \sum_{i \in S_k/\{\rho\}} u_i^m \right) \\
&\quad + \left(\sum_{j \in S_m} u_j^m - \sum_{j \in S_m} u_j^m \right) + \phi_{S_k}^{\rho} + \epsilon_{S_m}^{\tilde{\rho}} \\
&= u_\rho^k + 2 \cdot u_\rho^m + \phi_{S_k}^{\rho} + \epsilon_{S_m}^{\tilde{\rho}}
\end{aligned}
$$

□

To summarize, we have found that $W_\rho^{\mathbb{U}_{S_k}}$, which is the contribution of player ρ to its Swarm Utility Difference, consists of the sum of four terms, as shown in Theorem (13). The first term, u_ρ^k, is its utility resulted from intra-swarm interactions, the second term, $2 \cdot u_\rho^m$, is twice the utility resulted from inter-swarm interactions. The third term, $\phi^{\rho}_{S_k}$, is the utility accumulated by the other players in its own swarm, S_k, resulted from interactions with ρ, minus the alternative effect, which is the effect on other players in its own swarm when player ρ is absent. This term is equal to $\epsilon^{\rho}_{S_k} - \epsilon^{\tilde{\rho}}_{S_k}$, which is the difference between the effect of the presence and the absence of player ρ on its own swarm. The last term, $\epsilon^{\tilde{\rho}}_{S_m}$, is the effect of the absence of player ρ on the opponent swarm.

If known, the value of the $W_{\rho}^{\mathbb{U}_{S_k}}$ enables the players to estimate their own contribution to their swarm's utilities, thus determine the optimal action, from possible action-set.

The reader should note that in this chapter, there where no assumptions on individual and global utilities, except for the zero-sum game assumption for inter-swarm interactions. Therefore, such utilities can apply for many types of games and problems. In the following chapters, we make more specific assumptions to enable estimations and approximations of utilities by individual players, and therefore apply learning.

Chapter 4

Competitive Robot Swarms

In the previous chapter, we proposed a theoretical model of competitive multiple swarm game, where cooperative behavior occurs within the swarm and competitive behavior between swarms. The formulation makes no specific assumptions on either individual or global utilities, except for the zero-sum game assumption for individual inter-swarm interactions. In this chapter, we apply the theory to competitive robot swarms, where there are spatial interactions between individual robots during the performed game: collisions. To overcome the gap results from the partial information known to each robot, and the noisy local sensing, learning may be applied. To enable learning we propose a *Multi-Armed Bandit* (**MAB**) reinforcement learning model, and apply it to the theoretical model presented in the previous chapter. Section 4.1 presents the problem of spatial interactions applied to robots. Section 4.2 estimates the swarm utilities as a function of time. Section 4.3 further explores the contribution of an individual robot to the utility of the swarm as a function of time. And Section 4.4 proposes a learning model to be performed by each individual robot.

4.1 Spatial Interactions in *Robots*

We assume that when a robot detects another in its vicinity, it enters into a spatial interaction state, sometimes referred as *spatial conflict* or *collision*, marked with A. When entering such interaction, the robot stops executing its main task, and acts to address the spatial conflict according to a policy. Once the conflict is resolved, it returns to its main task, which is also referred as *program execution*, marked with P.

We assume that each robot has a swarm-identity that it is aware of, and is detectable by its surroundings. Thus, a robot is able to identify what is the swarm-identity of the robot it interacts with. The number of possible spatial-interaction types is the number of existing swarms in the field.

Figure (4.1) shows a typical sequence of robot behavior during a single run, which includes a sequence of interactions, separated by executing the main task.

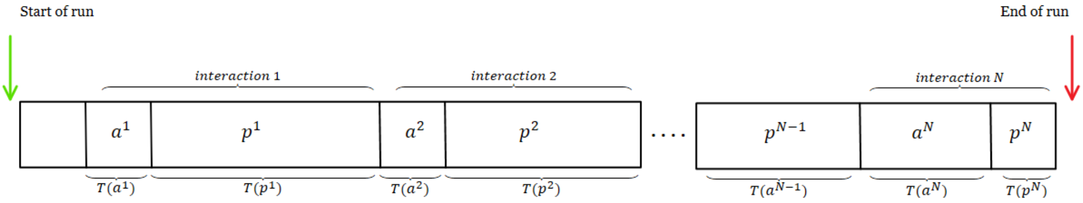
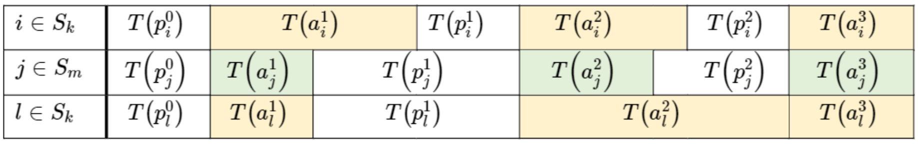


Figure 4.1: A typical time sequence of a single robot's interactions in a single run. Each interaction c composed of a response action a^c, followed by program execution p^c.

A single interaction c for a single agent i in the swarm can be represented as a tuple (a_i^c, p_i^c, s_i^c), where a_i^c is a collision-handling action from the set of possible actions $a_i^c \in \mathcal{A}$ when the robot is in state *Action* (A), and p_i^c is a program-execution action from the singleton set of actions $p_i^c \in \mathcal{P}$ when the robot is in state *Program* (P). The collision-state, denoted as $s_i^c \in \{1, .., K\}$ describes the collision type, which is the swarm identity of the interacted robot. We denote the set of actions taken by an individual robot i in each state (interaction type) as $\pi_i \in \mathcal{A}^K$. In other words, the policy defines an action suitable for every possible interaction type. We mark the related durations of the actions a and p in collision c as $T(a_i^c)$, $T(p_i^c)$ respectively. For simplicity of notation, when concentrating on a specific collision, we mark the interaction and program times as $T(a_i)$ and $T(p_i)$.



<table>
<tr><td>$i \in S_k$</td><td>$T(p_i^0)$</td><td colspan="2">$T(a_i^1)$</td><td>$T(p_i^1)$</td><td colspan="2">$T(a_i^2)$</td><td>$T(p_i^2)$</td><td>$T(a_i^3)$</td></tr>
<tr><td>$j \in S_m$</td><td>$T\left(p_j^0\right)$</td><td>$T\left(a_j^1\right)$</td><td colspan="2">$T\left(p_j^1\right)$</td><td>$T\left(a_j^2\right)$</td><td colspan="2">$T\left(p_j^2\right)$</td><td>$T\left(a_j^3\right)$</td></tr>
<tr><td>$l \in S_k$</td><td>$T(p_l^0)$</td><td>$T(a_l^1)$</td><td colspan="2">$T(p_l^1)$</td><td colspan="3">$T(a_l^2)$</td><td>$T(a_l^3)$</td></tr>
</table>

Figure 4.2: A representative time sequence of multi-player interactions. For each player, each interaction is composed of a response action, followed by program execution. Interactions of players from swarm S_k are marked in yellow, and of a player from S_m in green.

Fig. 4.2 demonstrates multiple interaction between players from different swarms. For each player, each interaction is composed of a response action, followed by program execution. The first and third rows correspond to two agents from the same swarm S_k.

In case that a new conflict is created during the resolution of the current conflict, a new resolution of the new conflict is started. The program time between these two conflicts will be 0. An example of this case is shown at the bottom row of Fig. 4.2, where a third collision occurs immediately after the second one for player $l \in S_k$.

4.2 Time as a Measure of Individual Player's Utility

Since robots in a system have limited sensing and communication capabilities, they are unable to know the utilities of other robots, even of those they interact with. Furthermore, each robot does not even know how its own action affects its own utility. The only information available to it is data from its own sensors and internal state information, which is accumulated and do not contain its full history. We build on Douchan et al. [18] and assume that the utility of a player from an interaction is a linear combination of $T(p), T(a)$. The challenge is to estimate the utility of a robot in a single interaction from $T(p), T(a)$, based both on the zero-sum assumption for inter-swarm interactions presented earlier, and on the previous works. More specifically, as described in the theoretical model, since there are different types of collisions, inter-swarm and intra-swarm, we assume different utility functions and values.

In order for a player to learn a policy in a game, it estimates a measure for its performance in different taken actions, i.e., its individual utility, defined in Section (3.1). The work [18] assumed that the individual utility increases with the program execution time, $T(p)$, and decreases during the time of interaction with other players, $T(a)$. We adopt this approach for intra-swarm interactions. For inter-swarm interactions, it can sometimes be more beneficial to invest the time in continuing the conflict rather than getting back to the program, i.e, interfere with the opponent, preventing it from execute its own main task (own program). Thus in this case, utility can *increase* with $T(p)$.

We now define α_i and β_i, used in the rest of this work. We denote the contribution of the program execution time, $T(p)$ for individual player, i, as $\beta_i \cdot T(p)$, where $\beta_i > 0$. Robots always operate in the same fashion during program execution time, and therefore any single player i, has a single parameter β_i. We also denote the contribution during interaction. Since the action during interaction time is a function of known actions, we denote the contribution of interaction time as a $\alpha_i(a_i, s_i)$, where a_i is the chosen action, and s_i is the state of the player during the interaction, i.e., the swarm identity of the colliding robot. To ease readability, we shall write $\alpha_i(a_i, s_i)$ as α_i. Therefore we can write this contribution as $-\alpha_i \cdot T(a)$, where $\alpha_i > 0$.

To preserve the generality of the approach, each player has its own weights α_i, β_i, which might differ from other players' weights, both players from its own swarm and from another competitive swarm. Furthermore, each player may use different inter- and intra-swarm weights to denote the different influence of those respective interactions.

We now propose terms for the individual utilities. We distinguish between the cases of single- and multi-player interactions, both for inter- and intra-swarm interactions. Based on Eq. (3.3) the individual utility, U_i, is the sum of all such interactions. We conclude by showing that the utilities satisfy the assumptions of Section 3.1.

4.2.1 Estimating Intra-Swarm Utility from Time

We now propose terms for individual utilities for the single and multiple-player intra-swarm interactions.

Proposition 14. *In the simple case of an intra-swarm collision between two players $i, j \in S_k$, the utility of player i from the intra-swarm interaction is taken to be:*

$$u^k_{i \in S_k} = \beta_i \cdot T(p_i) - \alpha_i \cdot T(a_i) \tag{4.1}$$

From Prop. 14 one can observe that the individual utility improves with longer program time, and shorter intra-swarm interaction. This is correct for all intra-swarm interaction, regardless of the exact values of α and β.

Proposition 15. *In the case of a multi-player intra-swarm interaction between robot $i \in S_k$ and n_k other robots from swarm S_k, we can split the interaction to n_k different collisions. The utility of player i from the intra-swarm interaction is proposed as:*

$$\begin{aligned} u^k_{i \in S_k} &= \sum_{1}^{n_k} (\beta_i \cdot T(p_i) - \alpha_i \cdot T(a_i)) \\ &= n_k \cdot (\beta_i \cdot T(p_i) - \alpha_i \cdot T(a_i)) \end{aligned} \tag{4.2}$$

4.2.2 Estimating Inter-Swarm Utility from Time

We now propose terms for individual utilities for the single and multiple-player inter-swarm interactions.

Proposition 16. *In the simple case of an inter-swarm interaction between two opponent robots $i \in S_k, j \in S_m$, where $k \neq m$, the utility of player i from the inter-swarm interaction is proposed:*

$$\begin{aligned} u^m_{i \in S_k} &= (\beta_i \cdot T(p_i) - \alpha_i \cdot T(a_i)) - (\beta_j \cdot T(p_j) - \alpha_j \cdot T(a_j)) \qquad (4.3) \\ &= (\beta_i \cdot T(p_i) - \beta_j \cdot T(p_j)) - (\alpha_i \cdot T(a_i) - \alpha_j \cdot T(a_j)) \end{aligned}$$

From Prop. 16, one can easily see that the utility of a player from an inter-swarm interaction, can either be negative or positive, and is influenced by the difference between the α and β values of the interacted players. Therefore, in some cases, during inter-swarm interactions, a better utility can be achieved from a longer interaction time with the opponent rather than in executing the main task.

Lemma 17. *Propositions* (14) *and* (16) *satisfy the zero-sum game assumption as in Eq.* (3.1)*, i.e.:*

$$u^m_{i \in S_k} + u^k_{j \in S_m} = 0$$

Proof. Replacing the indices of players i and j from the opponent swarms in Eq. (4.3), results with the utility of player $j \in S_m$ from the inter-swarm interaction:

$$u^k_{j \in S_m} = (\beta_j \cdot T(p_j) - \alpha_j \cdot T(a_j)) - (\beta_i \cdot T(p_i) - \alpha_i \cdot T(a_i)) \tag{4.4}$$

Summing Eq. (4.3) and (4.4):

$$\begin{aligned} u^m_{i \in S_k} + u^k_{j \in S_m} = & \\ &= ((\beta_i \cdot T(p_i) - \alpha_i \cdot T(a_i)) - (\beta_j \cdot T(p_j) - \alpha_j \cdot T(a_j))) \\ &\quad - ((\beta_j \cdot T(p_j) - \alpha_j \cdot T(a_j)) - (\beta_i \cdot T(p_i) - \alpha_i \cdot T(a_i))) \\ &= 0 \end{aligned}$$

□

Therefore, the zero-sum game assumption of Eq. (3.1) is satisfied using propositions (14) and (16).

We now expand a single inter-swarm interaction into a multi-player inter-swarm interaction.

Proposition 18. *In a case of a multi-player interaction between $i \in S_k$ and other n_m robots from swarm S_m, we can split the collision to n_m different collisions.*

Using Eq. (4.3) *to get:*

$$
\begin{aligned}
u^m_{i\in S_k} &= \sum_{\substack{j=1\\ j\in S_m}}^{n_m} ((\beta_i\cdot T(p_i)-\beta_j\cdot T(p_j))-(\alpha_i\cdot T(a_i)-\alpha_j\cdot T(a_j)))\\
&= \left(n_m\cdot\beta_i\cdot T(p_i)-\sum_{\substack{j=1\\ j\in S_m}}^{n_m}\beta_j\cdot T(p_j)\right)-\left(n_m\cdot\alpha_i\cdot T(a_i)-\sum_{\substack{j=1\\ j\in S_m}}^{n_m}\alpha_j\cdot T(a_j)\right)\\
&= n_m\cdot\left(\beta_i\cdot T(p_i)-\overline{\beta_m\cdot T(p_m)}\right)-n_m\cdot\left(\alpha_i\cdot T(a_i)-\overline{\alpha_m\cdot T(a_m)}\right)
\end{aligned}
\tag{4.5}
$$

where to simplify the notation, we define $\overline{\beta_m\cdot T(p_m)}$ *and* $\overline{\alpha_m\cdot T(a_m)}$ *as:*

$$
\begin{cases}
\overline{\beta_m\cdot T(p_m)} &= \frac{1}{n_m}\cdot\sum_{\substack{j=1\\ j\in S_m}}^{n_m}\beta_j\cdot T(p_j)\\
\overline{\alpha_m\cdot T(a_m)} &= \frac{1}{n_m}\cdot\sum_{\substack{j=1\\ j\in S_m}}^{n_m}\alpha_j\cdot T(a_j)
\end{cases}
\tag{4.6}
$$

Eq. (4.6) provides a simplified way for calculation of the individual utilities which is based on averages rather than on multiple individual values.

We now show that the inter-swarm utilities obtained by all participating players in a multi-player interaction also satisfies the zero-sum game assumption.

Lemma 19. *Eq.* (4.5) *of Proposition* (18) *satisfies the zero-sum game assumption of Eq.* (3.2), *i.e.:*

$$\sum_{i\in S_k} u^m_i + \sum_{j\in S_m} u^k_j = 0$$

Proof.

$$
\begin{aligned}
&\sum_{i\in S_k} u^m_i + \sum_{j\in S_m} u^k_j =\\
&= \sum_{i\in S_k}\left(n_m\cdot\beta_i\cdot T(p_i)-\sum_{j\in S_m}\beta_j\cdot T(p_j)-n_m\cdot\alpha_i\cdot T(a_i)+\sum_{j\in S_m}\alpha_j\cdot T(a_j)\right)
\end{aligned}
$$

$$
\begin{aligned}
&+\sum_{j\in S_m}\left(n_k\cdot\beta_j\cdot T(p_j)-\sum_{i\in S_k}\beta_i\cdot T(p_i)-n_k\cdot\alpha_j\cdot T(a_j)+\sum_{i\in S_k}\alpha_i\cdot T(a_i)\right)\\
&=n_m\cdot\sum_{i\in S_k}\beta_i\cdot T(p_i)-\sum_{i\in S_k}\sum_{j\in S_m}\beta_j\cdot T(p_j)-n_m\cdot\sum_{i\in S_k}\alpha_i\cdot T(a_i)+\sum_{i\in S_k}\sum_{j\in S_m}\alpha_j\cdot T(a_j)\\
&+n_k\cdot\sum_{j\in S_m}\beta_j\cdot T(p_j)-\sum_{j\in S_m}\sum_{i\in S_k}\beta_i\cdot T(p_i)-n_k\cdot\sum_{j\in S_m}\alpha_j\cdot T(a_j)+\sum_{j\in S_m}\sum_{i\in S_k}\alpha_i\cdot T(a_i)\\
&=n_m\cdot\sum_{i\in S_k}\beta_i\cdot T(p_i)-n_k\cdot\sum_{j\in S_m}\beta_j\cdot T(p_j)-n_m\cdot\sum_{i\in S_k}\cdot\alpha_i\cdot T(a_i)+n_k\cdot\sum_{j\in S_m}\alpha_j\cdot T(a_j)\\
&+n_k\cdot\sum_{j\in S_m}\beta_j\cdot T(p_j)-n_m\cdot\sum_{i\in S_k}\beta_i\cdot T(p_i)-n_k\cdot\sum_{j\in S_m}\alpha_j\cdot T(a_j)+n_m\cdot\sum_{i\in S_k}\alpha_i\cdot T(a_i)\\
&=0
\end{aligned}
$$

□

In this subsection we proposed a time-based utility estimation. We started with a simple case of interaction between two players from different swarms, and showed it to be a zero-sum game. We then expanded the utility to the case of single interaction with multiple opponents, concluded by Lemma 19, showing that zero sum game is valid at the swarm level.

4.2.3 Estimating Utility of Mixed Intra- and Inter-Swarm Interactions

The case of multi-player interaction from two different swarms, can also be formulated as follows.

Proposition 20. *In the complex case of a multi-player collision of n_k robots from swarm S_k and n_m robots from swarm S_m with player $i \in S_k$, we can split the collision to n_k intra-swarm collisions and n_m inter-swarm collisions. Using Eq.* (4.5) *and* (4.2)*, the utility of player i from this collision is:*

$$
\begin{aligned}
U_{i\in S_k}&=u_i^k+u_i^m \qquad (4.7)\\
&=n_k\cdot(\beta_i\cdot T(p_i)-\alpha_i\cdot T(a_i))\\
&\quad+n_m\cdot\left(\left(\beta_i\cdot T(p_i)-\overline{\beta_m\cdot T(p_m)}\right)-\left(\alpha_i\cdot T(a_i)-\overline{\alpha_m\cdot T(a_m)}\right)\right)
\end{aligned}
$$

Prop. (20) combines the previous terms proposed in this section, to specifically enable calculation of the individual utility of a player, based on its own times and parameters, and only average parameters for the other competitive swarm: $\overline{\alpha_m \cdot T(a_m)}$ and $\overline{\beta_m \cdot T(p_m)}$.

We can generalize the 2-swarm case into a K-swarm case, as follows:

Proposition 21. *In the general case of K swarms with $n_1, ..., n_K$ robots involved in a multiple-player collision:*

$$\begin{aligned} U_{i \in S_k} =& u_i^k + \sum_{m=1}^{K} u_i^m \\ =& n_k \cdot (\beta_i \cdot T(p_i) - \alpha_i \cdot T(a_i)) \\ &+ \sum_{\substack{m=1 \\ j \in S_m}}^{K} n_m \cdot \Big(\Big(\beta_i \cdot T(p_i) - \overline{\beta_m \cdot T(p_m)} \Big) - \Big(\alpha_i \cdot T(a_i) - \overline{\alpha_m \cdot T(a_m)} \Big) \Big) \end{aligned}$$

To conclude, we applied the assumptions of time as utility to the general theory proposed in the previous chapter, by defining individual weight parameters for interaction and program execution times. We proposed terms for the individual utilities for inter- and intra-swarm interactions, for both single- and multi-player interactions (Prop. (14), (15), (16), (18), (20) and (21)). We also showed the zero-sum game assumption is satisfied under the transformation to time as a proxy to utility (Lemmas (17) and (19)). We also concluded from Prop. (16) that for inter-swarm interaction, it can sometimes be more beneficial to interfere with an opponent player in performing its own task.

4.3 Using Time for WLU

We saw in previous works that WLU may be useful for a single player to estimate its contribution to the swarm. We now apply utility estimate from time (Prop. (20)) into $W_\rho^{\mathbb{U}_{S_k}}$, found in Section (3.3).

Proposition 22. *Using Eq.* (4.7) *and Theorem 13, we get:*

$$\begin{aligned} W_i^{\mathbb{U}_{S_k}} &= u_i^k + \phi_{S_k}^i + 2 \cdot u_i^m + \epsilon_{S_m}^{\bar{i}} \\ &= n_k \cdot (\beta_i \cdot T(p_i) - \alpha_i \cdot T(a_i)) + \phi_{S_k}^i + \epsilon_{S_m}^{\bar{i}} \\ &\quad + 2 \cdot n_m \cdot \Big((\beta_i \cdot T(p_i) - \alpha_i \cdot T(a_i)) - \Big(\overline{\beta_m \cdot T(p_m)} - \overline{\alpha_m \cdot T(a_m)} \Big) \Big) \end{aligned} \tag{4.8}$$

To evaluate $W_i^{\mathbb{U}_{S_k}}$, we can use expressions for the utilities u_i^k and u_i^m, of Eq. (4.8) in Section (4.2). We next concentrate on evaluating the terms $\phi_{S_k}^i$ and $\epsilon_{S_m}^{\tilde{i}}$.

As mentioned previously in this chapter, since robots in a system have limited sensing and communication capabilities, they are unable to know the utilities of other robots, even in the same interaction. The robots do not have the knowledge of the states of their surrounding robots, and therefore, they do not know how their own action influenced the program time of other robots. Hence, they need somehow to estimate their effect on other robots in every collision. Therefore, in order to calculate its own WLU, a robot must evaluate its impact on other robots, denoted earlier as ϕ, ϵ.

Proposition 23. *We propose the estimated effect of player $i \in S_k$ on player $j \in S_m$ in a specific collision when i is absent, marked as $\tilde{i}$, as follows:*

$$\epsilon_j^{\tilde{i}} = \beta_j \cdot (T(p_j) + T(a_j)) \tag{4.9}$$

$$\epsilon_{S_k}^{\tilde{i}} = n_k \cdot \overline{\beta_k} \cdot \Big(\overline{T(p_k)} + \overline{T(a_k)} \Big) = n_k \cdot \overline{\beta_k \cdot T(p_k)} + n_k \cdot \overline{\beta_k \cdot T(a_k)} \tag{4.10}$$

$$\epsilon_{S_m}^{\tilde{i}} = n_m \cdot \overline{\beta_m} \cdot \Big(\overline{T(p_m)} + \overline{T(a_m)} \Big) = n_m \cdot \overline{\beta_m \cdot T(p_m)} + n_m \cdot \overline{\beta_m \cdot T(a_m)} \tag{4.11}$$

Eq. (4.9) is an estimated utility a player $j \in S_m$ would have gotten in a specific collision if player $i \in S_k$ is absent. The idea is that if there was no conflict between the players, player j would be in program execution time instead of in interaction time. Therefore, it should add to the utility $\beta_j \cdot T(a_j)$ in the absence of i, instead of $\alpha_j \cdot T(a_j)$ in the presence of i.

Similarly to Eq. (4.9), equations (4.10) and (4.11) are the estimated utilities that swarms S_k and S_m with number of players n_k, n_m respectively, would have gotten in a specific collision if player $i \in S_k$ is absent, as they are multiplied by the average $\overline{\beta_m}$ instead of $\overline{\alpha_m}$.

Proposition 24. *We propose the estimated effect of player $i \in S_k$ on player $l \in S_k$ in a collision when i is present:*

$$\epsilon_l^i = \beta_l \cdot T(p_l) - \alpha_l \cdot T(a_l) \tag{4.12}$$

$$\epsilon_{S_k}^i = n_k \cdot \left(\overline{\beta_k \cdot T(p_k)} - \overline{\alpha_k \cdot T(a_k)}\right) \tag{4.13}$$

Due to Eq. (4.1), Eq. (4.12) is the utility got by player $l \in S_k$ in a specific collision when player $i \in S_k$ is present. Similarly, Eq. (4.13) is the utility got by swarm S_k, that counts n_k players.

We have found the effect of an individual player on its surrounding by applying the time based utility on the theory from the previous chapter. Our goal now is to find the value of a $W_i^{\mathbb{U}_{S_k}}$, the WLU of player $i \in S_k$ during a specific collision, using the estimated of utilities and effects in equations (4.8), (4.10),(4.11), and (4.13).

Proposition 25.

By assigning equations (4.10),(4.11),(4.13) *into Eq.* (4.8)*, the Wonderful Life Utility of player $i \in S_k$, during a collision is:*

$$\begin{aligned}
W_i^{\mathbb{U}_{S_k}} &= u_i^k + 2 \cdot u_i^m + \phi_{S_k}^i + \epsilon_{S_m}^{\tilde{i}} \\
&= \left(u_i^k + \epsilon_{S_k}^i - \epsilon_{S_k}^{\tilde{i}}\right) + \left(2 \cdot u_i^m + \epsilon_{S_m}^{\tilde{i}}\right) \\
&= \left(n_k \cdot (\beta_i \cdot T(p_i) - \alpha_i \cdot T(a_i)) + \epsilon_{S_k}^i - \epsilon_{S_k}^{\tilde{i}}\right) \\
&\quad + \left(2 \cdot n_m \cdot \left((\beta_i \cdot T(p_i) - \alpha_i \cdot T(a_i)) - \left(\overline{\beta_m \cdot T(p_m)} - \overline{\alpha_m \cdot T(a_m)}\right)\right) + \epsilon_{S_m}^{\tilde{i}}\right) \\
&= n_k \cdot (\beta_i \cdot T(p_i) - \alpha_i \cdot T(a_i)) + n_k \cdot \left(\overline{\beta_k \cdot T(p_k)} - \overline{\alpha_k \cdot T(a_k)}\right) \\
&\quad - n_k \cdot \overline{\beta_k \cdot T(p_k)} - n_k \cdot \overline{\beta_k \cdot T(a_k)} + 2 \cdot n_m \cdot (\beta_i \cdot T(p_i) - \alpha_i \cdot T(a_i)) \\
&\quad - 2 \cdot n_m \cdot \left(\overline{\beta_m \cdot T(p_m)} - \overline{\alpha_m \cdot T(a_m)}\right) + n_m \cdot \overline{\beta_m \cdot T(p_m)} + n_m \cdot \overline{\beta_m \cdot T(a_m)} \\
&= n_k \cdot \beta_i \cdot T(p_i) - n_k \cdot \alpha_i \cdot T(a_i) + n_k \cdot \overline{\beta_k \cdot T(p_k)} - n_k \cdot \overline{\alpha_k \cdot T(a_k)} \\
&\quad - n_k \cdot \overline{\beta_k \cdot T(p_k)} - n_k \cdot \overline{\beta_k \cdot T(a_k)} + 2 \cdot n_m \cdot \beta_i \cdot T(p_i) - 2 \cdot n_m \cdot \alpha_i \cdot T(a_i)
\end{aligned}$$

$$
\begin{aligned}
&- 2 \cdot n_m \cdot \overline{\beta_m \cdot T(p_m)} + 2 \cdot n_m \cdot \overline{\alpha_m \cdot T(a_m)} + n_m \cdot \overline{\beta_m \cdot T(p_m)} + n_m \cdot \overline{\beta_m \cdot T(a_m)} \\
=\; &(n_k \cdot \beta_i + 2 \cdot n_m \cdot \beta_i) \cdot T(p_i) - (2 \cdot n_m \cdot \alpha_i + n_k \cdot \alpha_i) \cdot T(a_i) \qquad (4.14) \\
&- n_k \cdot \left(\overline{\alpha_k \cdot T(a_k)} + \overline{\beta_k \cdot T(a_k)}\right) - n_m \left(2 \cdot \overline{\alpha_m \cdot T(a_m)} + \overline{\beta_m \cdot T(a_m)} - \overline{\beta_m \cdot T(p_m)}\right)
\end{aligned}
$$

So far we have quantified WLU based on the program and action times of a single robot in a multi-swarm competitive game, using the defined weights α and β and the number of interacting robots, while taking into account all swarm-identities.

4.4 Learning in Multi-Swarm Competition

To overcome the gap in knowledge, resulting from the partial information known to each robot, we apply learning. The learning process must be individual, and each robot should estimate parameters rather than use the true but unknown values during the learning process. Rather, each robot should use known information to evaluate its actions, thus learn a specific policy π_i. The goal is to find a policy, π_i^*, which maximizes $W_i^{\mathbb{U}_{S_k}}$ and thus the swarms' policies $\pi_{S_k}^*$ that maximize the Swarm Utility Difference $\mathbb{U}_{S_k}$.

In the following section we define and formulate the proposed learning model which is comprised of a reward function, a proposed estimation and evaluation, and the learning process. We relate it to the WLU of player i from a single interaction, as the reward obtained after performing an action, and mark it as $R(T(a_i), T(p_i))$, and use it to select optimal actions.

4.4.1 Evaluation and Selection of Actions

The simplest reinforcement-learning problem is known as the Multi-Armed Bandit (**MAB**) problem [23]. In MAB, the agent is in a room with multiple gambling machines. At each time-step the agent pulls the arm of one of the machines and receives a reward. The agent's purpose is to maximize its total reward over a sequence of trials. Usually, each arm is assumed to have a different distribution of rewards, therefore, the goal is to find the arm with the best expected return as early as possible, and then to keep gambling using that arm.

Thus, in MAB problem, the agent estimates each action's value, which is the the action's expected or mean reward, marked as $q(a) = \mathbb{E}[R]$. We denote the estimated value of action a for player $i,$ as $Q_i(a) \approx q(a)$. We use a common estimation method in MAB problems called *Sample Average* [54]. This method estimates the value of an action a by dividing the total reward achieved by the agent when action a was taken in previous steps, by the total time that a was taken. Hence, each robot's estimation of its value of action a is:

$$Q_i(a) = \frac{\sum_{c=1}^{N(a)} R\left(T(a_i^c), T(p_i^c)\right)}{\sum_{c=1}^{N(a)} \left(T(a_i^c) + T(p_i^c)\right)} \tag{4.15}$$

where $N(a)$ is the number of times action a was taken by the robot so far.

Techniques from MAB problems have been used to deal with the exploration-exploitation trade-off. In other words, the robot can take one of $|\mathcal{A}|$ actions, and can evaluate its actions using a reward function, R. At any such decision point, the choice of the action can be greedy, i.e., by taking the action that has the best value at that point. However, this policy is not optimal when reward are stochastic, as in our case, since non-greedy action might found to be better in the longer term. Therefore, it is valuable to use a known action selection method named Upper Confidence Bound (**UCB**), where the uncertainty value of an action is considered during the decision process [54]. The UCB algorithm [4] guarantees low regret under certain conditions. UCB uses the principle of optimism in the face of uncertainty to select its actions, by assuming an optimistic guess on the expected rewards. The use of MAB and UCB in the proposed model is described in this section.

The idea of UCB action selection is that the square root term is a measure of the uncertainty or variance in the estimate of a's value:

$$a^* = \underset{a}{argmax} \left[Q\left(a\right) + \sqrt{\frac{\gamma \cdot \log_e N}{N\left(a\right)}} \right] \tag{4.16}$$

where γ is a hyper-parameter affects exploration rate of UCB, and N is the total number of collisions, regardless of the action selection, i.e.,

$$N = \sum_{a \in \mathcal{A}} N(a)$$

A typical game that contains learning is usually partitioned into two parts, the first is when learning or training of a system occurs, and the second is operating the trained model and measuring its success. After the training period has ended, each individual robot has its own learned policy, π_i^*, consists of a set of actions for each opponent swarm. This policy is used during the evaluation phase. When the learning process has ended, the action selected for each swarm identity is the one maximizes the value of $Q(a)$. The set of all actions selected, creates the policy π_i^*.

4.4.2 The Learning Process

We now use all presented parts to build a learning procedure, to find a suitable policy for each individual player, that will eventually contribute to better performance of its whole swarm. The learning process contains the following steps, described in Algorithm 1.

To conclude this chapter, we concentrated on the domain of competitive robot swarms. We made some specific assumptions to enable estimations and approximations of utilities by individual players, to enable reinforcement learning. We proposed a learning model and applied it to the theoretic model presented in the previous chapter. However, we still do not assume specific operations or tasks; one possible example for such task is, multi-swarm competitive foraging, presented in the next chapters, and validates the proposed learning model.

Given that a robot i is about to interact with another player, the robot executes the following procedure:

1. Identify the opponent robot and immediate surroundings and environment.
2. Choose action a to react the to collision, using Eq. (4.16).
3. Perform the action a to respond to the spatial conflict, and obtain the interaction time $T(a_i)$.
4. Return to program execution up until next collision.
5. When the robot is about to collide once again the robot:
    (a) Obtains the program time $T(p_i)$.
    (b) Calculates the reward approximation, using Eq. (5.4), and its parameters.
6. Update Q(a) using the rule in Eq. (4.15).
7. Return to step 1 until stop criterion has been reached.
8. Select the best evaluated policy, by selecting the actions with the maximal value of $Q(a)$ for each swarm identity.

Algorithm 1: MAB Learning Process using UCB

Chapter 5

Competitive Multi-Swarm Robot Foraging

We apply the theoretical model presented in the previous chapters to robots. We take foraging as a basis for a robot multi-swarm competition, and define policies and methods that are used by the swarms during the competition. Such competition involves multiple swarms that compete over the same limited resources, and the goal is to collect more than any other swarm. We examine both baseline cases where no learning is applied, and cases where a learning process is performed by robots, based on the model presented and discussed in the previous chapters, including the assumption that individuals are aware of their swarm identities, and those of others.

In Section 5.1 we present a 3D physics-based simulation environment in which we apply the model robots. In Section 5.2 we apply the model to competitive foraging task in robots, including specific actions, policies, and logic. As the true utility cannot be directly calculated by any individual, we show in Section 5.3 how robots should approximate and learn their individual actions, based on the limited knowledge and on the models presented previously.

5.1 Simulated Robots and Competitive Foraging Environment

We use a multi-robot simulator named ARGoS [41] for simulations experiments. ARGoS is designed to simulate complex experiments involving large swarms of robots of various types. The type of robot we use for the simulation experiments is called Krembot, made by Robotican. The Krembot is a robot with relatively limited sensing, processing and memory capabilities. It is a cylindrical-shaped robot with height of 10.5 cm and diameter of 6.5 cm. Each robot has eight RGB LEDs, eight pressure sensors, eight RGB and light sensors, and eight proximity sensors. The robots cannot communicate with each other via WiFi or other digital communications.

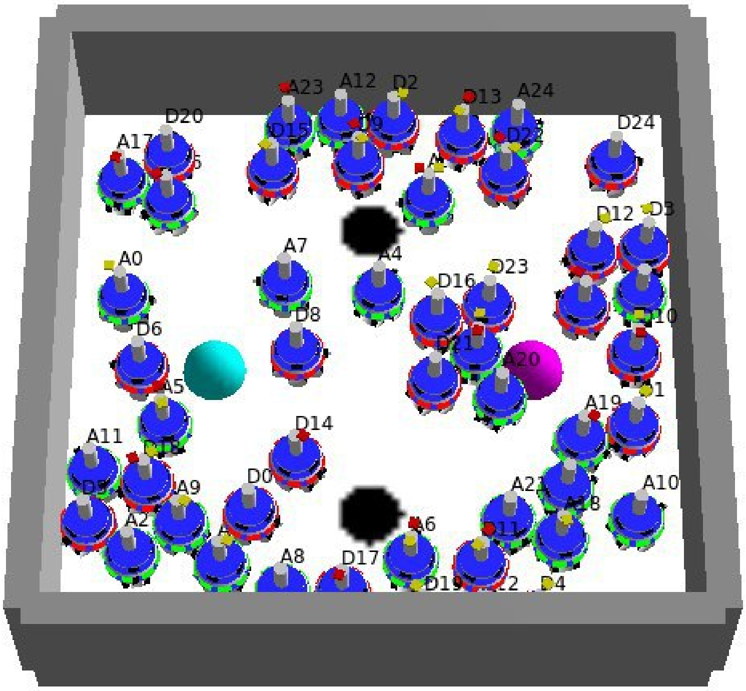


Figure 5.1: Simulated arena of two swarms of 25 robots each, marked with green and red lights.

Each robot has a predefined swarm identity, which it is aware of, and is displayed through its LEDs' color. Therefore, a robot can identify whether another robot belongs to its own swarm. The Krembots have no localization capabilities, and are unable to map their environment, or plan paths.

Figure 5.1 shows an example of simulated arena with two swarms containing 25 robots each, marked with green or red lights, and their respective home bases colored cyan and magenta. The black circles on the floor are the puck distribution stations, which have predefined supplies of pucks.

Using the ARGoS simulator, we are able to experiment with various simulation parameters. These include types of run constraints (e.g. time or resource limits); number of available resources such as total deposited pucks, supply stations and home bases; conflict identification parameters (e.g. which sensor and what distance are defining a spatial conflict); interaction response parameters and strategy; and global environment parameters such as size of arena, number of robots in each swarm, number of swarms.

In most experiments, we simulated two different swarms, where the swarms' size is equal and varied from three to 25 robots per swarm, and is unknown to the individual agent. In some other experiments we use three swarms or unequal swarm sizes. The arena's size is 2x2 meters in all simulations, includes two supply stations where the robots collect pucks from, and two home bases, one for each swarm, where the loaded robots deposit their pucks. Every scenario was tested for 20 runs with different random seeds. In each simulation run, we measure how many pucks each swarm brought in total to its home base. In a puck-limited scenario, the run ends when the total number of deposited pucks is 50. In a time-limited scenario the run ends after 10 minutes. In scenarios where learning is involved, the robots learn their optimal policy during a 50 minute game, and this learning period is not measured towards the goal. The results are measured when each robot uses the best policy it evaluated during the learning process.

5.2 Robots Carrying Out Competitive Foraging

To test the proposed models and algorithms, we use a variant of multi-robot foraging, where each swarm has its own home base which is marked by its color.

At any time, every robot can be in either the state of program execution (P), or the state of conflict resolution (A), where the robot selects between actions based on its (fixed or learned) policy.

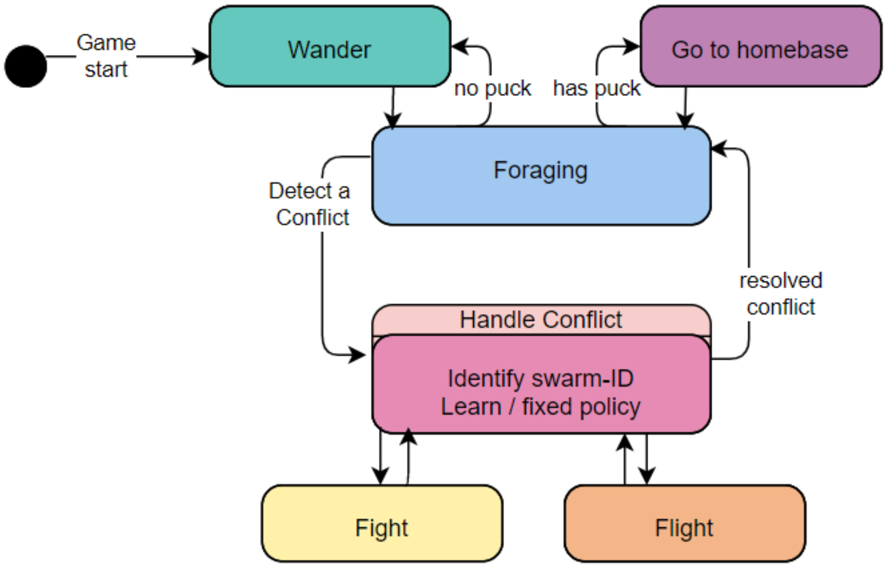


Figure 5.2: Competitive Foraging States Overview

The robots control process is overviewed in Figure 5.2. When not having a puck, the robot randomly wanders, looking for a puck in the arena. If a puck was found, the robot searches and moves towards its home base using its RGB sensors, to put down the puck. In both states, if a robot recognizes another robot from a distance smaller or equal to a predefined value from its front sensors, it responds to the spatial conflict with a chosen action. It may use different actions depending on both robots' swarm identities.

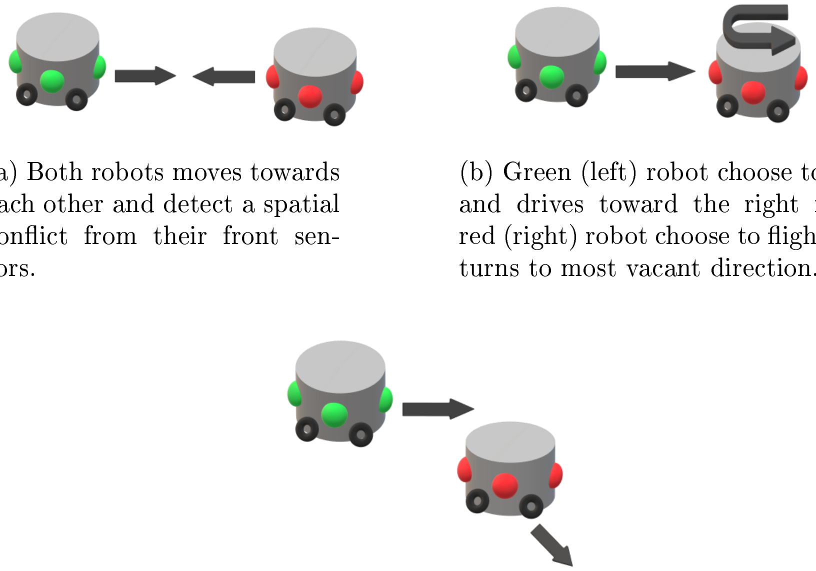
Therefore, during the foraging wander state the robots randomly searching for a station. During the return to home base state, the robots are searching their home base direction using their RGB sensors.

When responding to a spatial conflict or collision when it is detected, we define a set of two possible actions, and name them *fight* and *flight*.

- In the *flight* method, the robot change its direction to the most vacant direction for a per-determined amount of time, to resolve the spatial conflict as quickly as possible.

- In the *fight* method, the robot drives towards the other robot for a given amount of time, in order to interfere with the other robot's operation.

Figure 5.3 shows an illustration of an interaction between two robots from different swarms, as marked by their LED's colors. The illustration the detection of a spatial conflict (Fig. 5.3a), the fight and flight methods (Fig. 5.3b), and the resolution of the conflict (Fig. 5.3c). Note that if both robots would have chosen to fight with each other, there would have been a spatial collision between them.



(a) Both robots moves towards each other and detect a spatial conflict from their front sensors.

(b) Green (left) robot choose to fight and drives toward the right robot, red (right) robot choose to flight, and turns to most vacant direction.

(c) The conflict is resolved, red robot wanders towards a vacant direction, green robot wanders in the same direction.

Figure 5.3: Illustration of an example of *fight* and *flight* methods.

According to the different methods we define several robots' policies.

- The *Avoid* policy, always avoids collisions, i.e., use the *flight* method when colliding with a robot, regardless of its swarm identity.

- The *Attack* policy, always use the *fight* method in inter-swarm collision, and always use the *flight* method in intra-swarm collision.

- The *Aggressive* policy, always use the *fight* method when colliding with a robot, regardless of its swarm identity.
- We call a robot that learns the best policy in a given environment a *Learner*.

We use the *flight* method for intra-swarm interactions since this was the preferred action for intra-swarm interaction, as shown in Subsection 6.2.3.

Since communication between individual robots is forbidden, each robot can only learn based on its own knowledge of the environment and its own history of collisions and choices. Therefore, both a swarm where all agents use the *Attack* policy, and swarm where all of agents use the *Avoid* policy are homogeneous swarms. However, a swarm where all agents learn their own best policy could become a heterogeneous swarm at the end of the learning process, as described in the previous chapter.

5.3 Learning in Swarm Robots

The reward function is used by an individual player to estimate its actions based solely on the parameters that are known to it, as described in a previous chapter. This requires estimating several unknowns. In order to approximate the reward function, the individual robot should estimate over time, the contribution of both executing its foraging task, and interacting with others.

In the previous chapter we denoted the contribution of the program execution time, $T(p)$ for individual player's reward function as $\beta \cdot T(p)$, and the contribution of the collision duration $T(a)$, as $\alpha(a,s) \cdot T(a)$, which depends on the swarm identity of the colliding robot, s, and the action taken, a (either *fight* or *flight*). Based on Eq. (4.14) we discriminated between α, β of a player i, its opponent j and a member of its team, l , as $\alpha_i, \alpha_j, \alpha_l, \beta_i, \beta_j, \beta_l$ respectively.

Using the assumption of previous work [18], that the whole system enters conflict-resolution state at the same time (i.e., collisions are mutual to all), we can write:

$$\begin{cases} T(p) = & T(p_i) = \overline{T(p_m)} = \overline{T(p_k)} \\ T(a) = & T(a_i) = \overline{T(a_m)} = \overline{T(a_k)} \end{cases}$$

Under this assumption, the terms from Eq. 4.6 can be rewritten as follows:

$$\begin{cases} \overline{\beta_m \cdot T(p_m)} & = \frac{1}{n_m} \cdot \sum_{\substack{j=1 \\ j \in S_m}}^{n_m} \beta_j \cdot T(p_j) = T(p) \cdot \left(\frac{1}{n_m} \cdot \sum_{\substack{j=1 \\ j \in S_m}}^{n_m} \beta_j \right) = \overline{\beta_m} \cdot T(p) \\ \overline{\alpha_m \cdot T(a_m)} & = \frac{1}{n_m} \cdot \sum_{\substack{j=1 \\ j \in S_m}}^{n_m} \alpha_j \cdot T(a_j) = T(a) \cdot \left(\frac{1}{n_m} \cdot \sum_{\substack{j=1 \\ j \in S_m}}^{n_m} \alpha_j \right) = \overline{\alpha_m} \cdot T(a) \end{cases}$$

By assigning the above equations to Eq. (4.14), we get the WLU value for player $i \in S_k$:

$$\begin{aligned} W_i^{\mathbb{U}_{S_k}} &= (n_k \cdot \beta_i + 2 \cdot n_m \cdot \beta_i) \cdot T(p) - (2 \cdot n_m \cdot \alpha_i + n_k \cdot \alpha_i) \cdot T(a) \\ &- n_k \cdot \left(\overline{\alpha_k} \cdot T(a) + \overline{\beta_k} \cdot T(a) \right) - n_m \left(2 \cdot \overline{\alpha_m} \cdot T(a) + \overline{\beta_m} \cdot T(a) - \overline{\beta_m} \cdot T(p) \right) \\ &= \left(n_k \cdot \beta_i + 2 \cdot n_m \cdot \beta_i - n_m \cdot \overline{\beta_m} \right) \cdot T(p) + \left(n_m \cdot \overline{\beta_m} - n_k \cdot \overline{\beta_k} \right) \cdot T(a) \\ &+ (-2 \cdot n_m \cdot \alpha_i - n_k \cdot \alpha_i - n_k \cdot \overline{\alpha_k} + 2 \cdot n_m \cdot \overline{\alpha_m}) \cdot T(a) \end{aligned} \tag{5.1}$$

We denote $n = n_k + n_m$, where n_k is the number of players from swarm k, n_m is the number of players from swarm m, and n is their sum, which is the constant total number of players, defined for each game.

Dividing by n, we get:

$$\begin{aligned} \frac{W_i^{\mathbb{U}_{S_k}}}{n} =& \frac{\left(\beta_i \cdot (n_k + n_m) + n_m \cdot \left(\beta_i - \overline{\beta_m} \right) \right)}{n} \cdot T(p) + \frac{\left(n_m \cdot \overline{\beta_m} - n_k \cdot \overline{\beta_k} \right)}{n} \cdot T(a) \\ +& \frac{(2 \cdot n_m \cdot \overline{\alpha_m} - n_k \cdot \alpha_i - n_k \cdot \overline{\alpha_k} - 2 \cdot n_m \cdot \alpha_i)}{n} \cdot T(a) \\ =& \left(\beta_i + \frac{n_m \cdot \left(\beta_i - \overline{\beta_m} \right)}{n} \right) \cdot T(p) \\ +& \left(\frac{\left(n_m \cdot \overline{\beta_m} - n_k \cdot \overline{\beta_k} \right)}{n} + \frac{(2 \cdot n_m \cdot \overline{\alpha_m} - n_k \cdot \alpha_i - n_k \cdot \overline{\alpha_k} - 2 \cdot n_m \cdot \alpha_i)}{n} \right) \cdot T(a) \end{aligned} \tag{5.2}$$

To simplify the notation, we define two terms and rewrite Eq. (5.2):

$$\frac{W_i^{\mathbb{U}_{S_k}}}{n} = \frac{\Gamma(\beta, n_k, n_m)}{n} \cdot T(p) + \frac{\Delta(\alpha, \beta, n_k, n_m)}{n} \cdot T(a) \tag{5.3}$$

Since robots in the arena have limited knowledge, they are not aware of the exact values of the above parameters, excluding their own program and interaction times, $T(a)$ and $T(p)$. From Prop. 14 and 16, one optional approximation for a single robot should include a sign change depending on the interaction type and action taken, and some estimate to the number of players from every swarm. Therefore, based on Eq. (5.3), we use the following reward, R, which is the approximated WLU of player i:

$$R = \frac{W_i^{\mathbb{U}_{S_k}}}{n} \approx \frac{\Gamma(\beta, n_i^{'})}{n'} \cdot T(p) + \left(\frac{n_i^{'}}{n'} \cdot \sigma_i \cdot |\Delta(\alpha, \beta)|\right) \cdot T(a) \tag{5.4}$$

where σ_i, n' and $n_i^{'}$ are explained below.

The actual number of robots affected by a robot's action, which is marked n, is unknown to the robot. In Eq. (5.4) we approximated n, n_k and n_m of Eq. (5.3) by sensor-based estimations for the number of affected robots in the current interaction, marked n' and $n_i^{'}$. n' is the maximum number of surrounding robots sensed, which for Krembot is eight, and $n_i^{'}$ is the number of the robot's sensors that are activated due to proximity of objects, which is the number of surrounding robots.

The sign function $\sigma_i(a, s)$,which depends on action and interaction type, reflects part of the influence of a specific action a, taken by robot i, on its surroundings. We assume that the contribution on the reward, depends on the direction of movement. When driving forward towards a friend robot, or backwards from a friend robot, may have opposite signs. Similarly, when moving in the same directions towards a friend or towards an opponent. The same should apply to other action types. We therefore approximate σ as:

$$\sigma_i(a, s) = f_i(s) \cdot d_i(a) \tag{5.5}$$

Where $d_i(a)$ is the direction of driving in selected action a during interaction,

$$d_i(a) = \begin{cases} 1 & forward \\ -1 & backward \end{cases}$$

and $f_i(s)$ is the interaction type, i.e, whether the interaction is inter- or intra-swarm (friend or foe).

$$f_i(s) = \begin{cases} 1 & inter - swarm\ collision \\ -1 & intra - swarm\ collision \end{cases}$$

Note that for two collision with identical selected action type, the signs of σ may be equal or opposite, based on the identity of the colliding robots.

We will distinguish between two cases. The first case is where all swarms act identically during the program execution time, i.e, foraging state. This case is formulated and evaluated empirically in Chapter 6. The second case is where one swarm has some advantage over the others in its program execution. This case is formulated in and evaluated in Chapter 7.

Chapter 6

Competitive Foraging with Symmetric Swarms

In this chapter, we concentrate on the case of competing swarms consist of robots with identical behavior during the program execution state (but not necessarily during conflict resolution), the robots behave exactly the same when searching for pucks and moving to their respective homebases. We thus explore the behavior during interaction, which may be either fixed or learned. In the next chapter, we further elaborate on the cases of non-identical program executing behavior for different swarms, and in particular when one of the swarm has some advantage over the other, and even a case of three-swarms competition.

In Section 6.1, we specifically present the reward function used for the cases of identical competing swarms. In Section 6.2 we show the experiment results of baseline cases using the proposed models, both for fixed and learned policies. In Section 6.3 we explore the performance of the learning model in different environments and hyper parameters, thus experimentally justify assumptions made as a basis for the learning model.

6.1 The Reward Function for the Case of Identical Swarms

The first scenario we analyze is the case where both swarms execute identically during program time. Since the behavior is identical, we assume the contribution of the program execution time for each robot, regardless of its swarm, is also identical. Therefore, we can assign

$$\beta_i = \beta_j = \overline{\beta_k} = \overline{\beta_m} = \beta$$

into Eq. (5.2), to get the reward of player i during a collision:

$$\begin{aligned} \frac{W_i^{\mathbb{U}_{S_k}}}{n} &= \left(\frac{(n_k + n_m)}{n} \cdot \beta\right) \cdot T(p) + \left(\frac{(n_m - n_k)}{n} \cdot \beta\right) \cdot T(a) \\ &+ \left(\frac{2 \cdot n_m \cdot \overline{\alpha_m} - n_k \cdot \overline{\alpha_k} - (2 \cdot n_m + n_k) \cdot \alpha_i}{n}\right) \cdot T(a) \\ &= \beta \cdot T(p) + \left(\frac{(n_m - n_k) \cdot \beta}{n} + \frac{2 \cdot n_m \cdot (\overline{\alpha_m} - \alpha_i) - n_k \cdot (\overline{\alpha_k} + \alpha_i)}{n}\right) \cdot T(a) \end{aligned} \tag{6.1}$$

From the above formula, one can observe that the program time $T(p)$ is multiplied by β, a positive parameter, which is the contribution of the program execution time, and the collision time $T(a)$ is multiplied by a term which might be either positive or negative. From Eq. (6.1) we can observe that the sign depends on the different α values, which are unknown to the participating players.

Therefore we can rewrite Eq. (5.4), using Eq. (6.1), to get the approximated reward for the identical competing swarms case:

$$R = \frac{W_i^{\mathbb{U}_{S_k}}}{n} \approx \beta \cdot T(p) + \left(\frac{n_i'}{n'} \cdot \sigma_i \cdot |\Delta(\alpha, \beta)|\right) \cdot T(a) \tag{6.2}$$

Since β is a positive constant, we can normalize the above equation, dividing by β. We use this normalized reward in the learning process of the experiments in the rest of this chapter, where different values are assigned to $\Delta(\alpha, \beta)$, and the effect on the results is demonstrated and discussed.

6.2 Baseline Cases

The following experiments are used as a baseline for the rest of this chapter. They first examine the case where all swarms have a fixed, predefined policy used by each of them during competition. We then experiment the baseline case of applying learning for inter-swarm interactions, when competing a non-learning swarm that uses a fixed predefined policy. Finally, we apply learning for intra-swarm interaction during competition, and applied simultaneous learning for both inter- and intra-swarm interaction during multi-swarm competition.

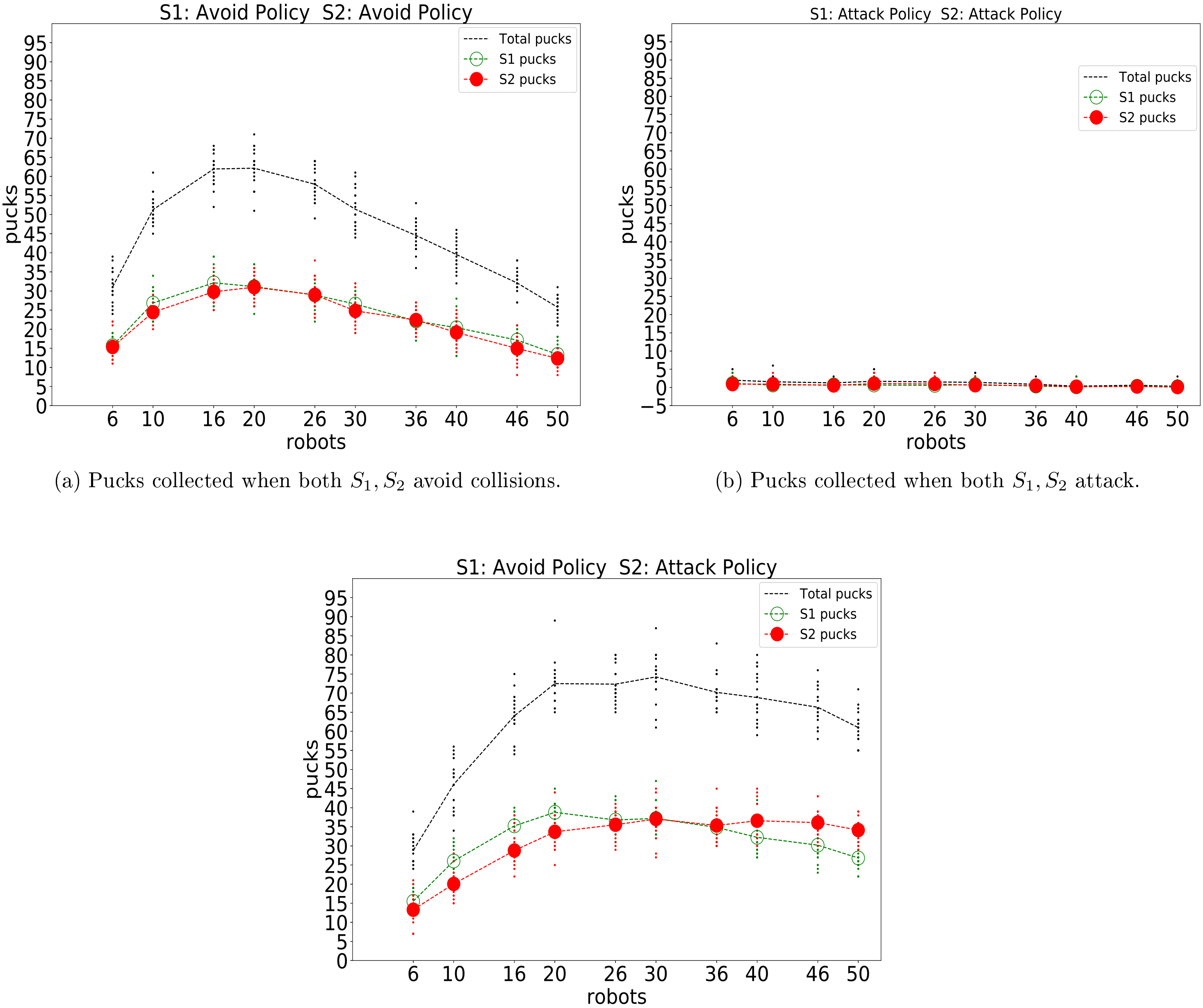
A description of the simulation environment and details for the multiple runs in various densities are detailed in Section 5.1.

6.2.1 Experiments with Fixed-Policy Competition

Each figure below summarizes a set of runs for a specific scenario. The horizontal axis shows the total number of robots in the arena, from both equally sized swarms. The vertical axis shows the number of pucks deposited to homebases, sometimes referred to as the score, or number of collected pucks. Swarm S_1 is marked in a green dashed line and unfilled circles. Swarm S_2 is marked in red dashed line and filled circles. The results can be grouped based on the stop condition for the experiment, either time-limited or puck-limited.

Figure 6.1 shows simulation results of two swarms in time-limited experiments of 10 minutes, when all swarms use fixed predefined policies and no learning is applied. The robots are tested in a fixed size arena. The number of robots in each swarm is varied from three to 25 robots, and is unknown to the swarm members.

In Figure 6.1a, both swarms use *Avoid* policy: use *flight* method in every interaction, meaning similar actions to all robots, regardless of swarm identities. Therefore, there is no actual competition. The scores are similar as expected, and the total number of pucks is directly influenced by the robot density in the arena. In low densities, the collected number of pucks is increasing when the swarms' size is increased, due to a better coverage of the field, but still small number of spatial interactions between robots. In high densities, the number of collected pucks is decreasing when the swarms' size is increased, due to a higher rate of interactions between robots. These results of Figure 6.1a, are consistent with previous work [18], since all robots can be considered as a single swarm.



(a) Pucks collected when both S_1, S_2 avoid collisions.

(b) Pucks collected when both S_1, S_2 attack.

(c) Pucks collected when S_1 avoids and S_2 attacks .

Figure 6.1: Pucks collected by two competitive swarms in a 10 minutes limited game with different densities and fixed swarms-policies.

Figure 6.1b shows the results of two swarms use *Attack* policy: use *fight* method in inter-swarm interactions, and use *flight* method in intra-swarm interactions. The score resulted from these runs are extremely low compared to all other cases. In this case, all robots are typically occupied continuously with repeated interactions with opponent robots, and are less available to collect pucks.

Figure 6.1c shows the first case of two competitive swarms with different interaction policies: swarm S_1 uses *Avoid* policy, and swarm S_2 uses *Attack* policy. Score results are better for *Avoid* policy in low densities, and for *Attack* policy in high densities. Notice that competition seems to contributes to increase the total number of pucks collected by all swarms, when compared to 6.1a.

To continue the exploration of the baseline, we repeat the last case shown in Fig. 6.1c, but this time, shown in Fig. 6.2, the runs where limited by the total number of pucks collected. The same insights seen before are reflected in the current case.

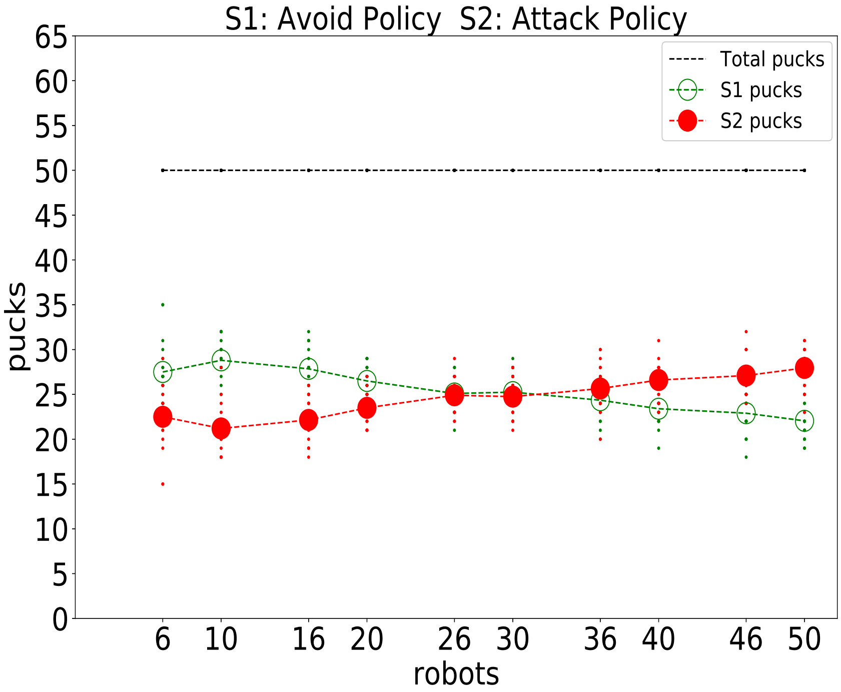


Figure 6.2: Number of pucks collected by two competitive swarms in a 50 pucks limited game, S_1, green, uses Avoid policy, S_2, red, uses Attack policy.

The results shown in Figures 6.1c and 6.2 comply with the initial intuition for lower densities. That is since if more time is invested in foraging rather than in interaction with the opponent, more pucks are likely to be collected for the swarm. However, the results at higher densities are quite surprising, since although most of the time for attacking swarm is devoted to interaction with the opponents, the attacking swarm still collect more pucks than the avoiding swarm.

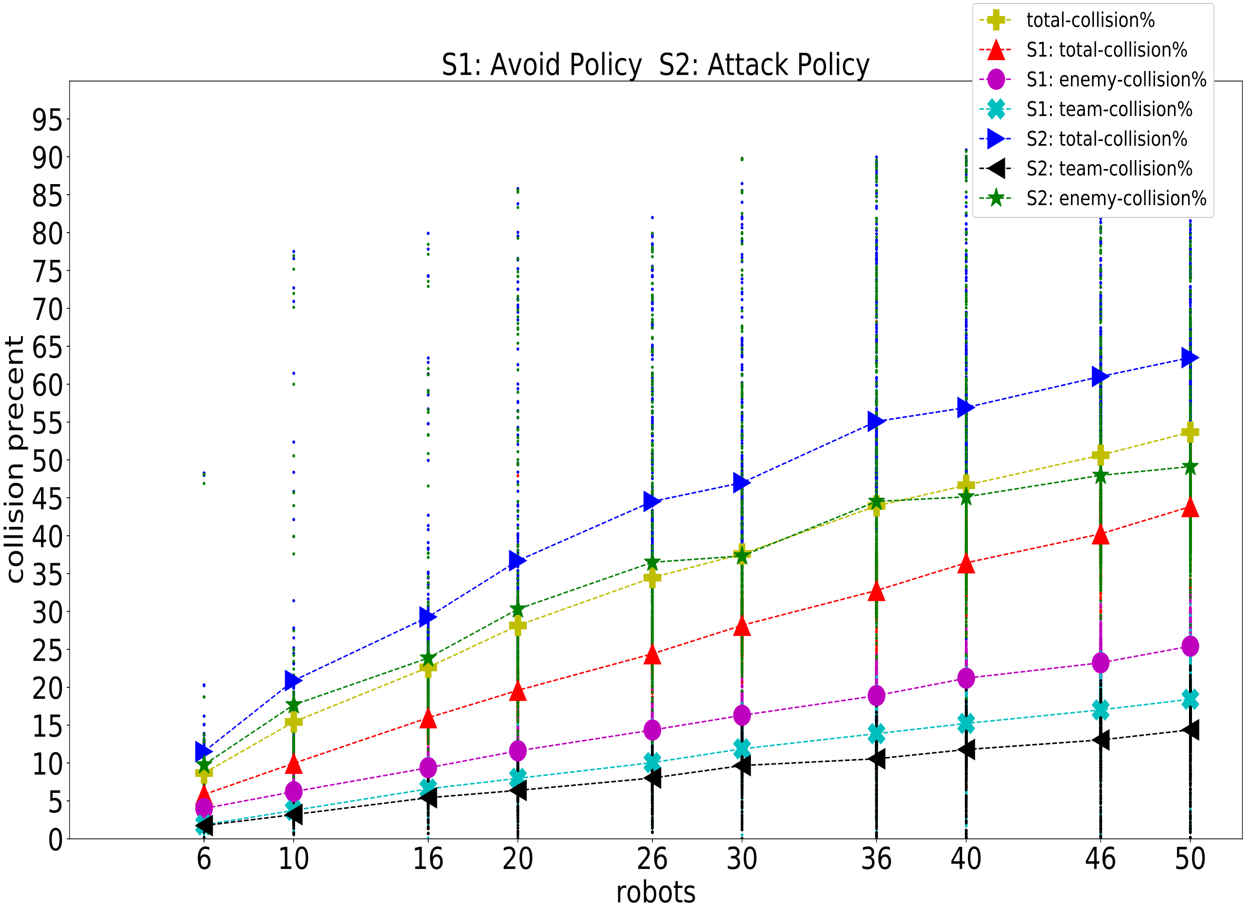


Figure 6.3: Percentage of inter-, intra-, and total-collision time for both swarms of experiment shown in Fig. 6.2.

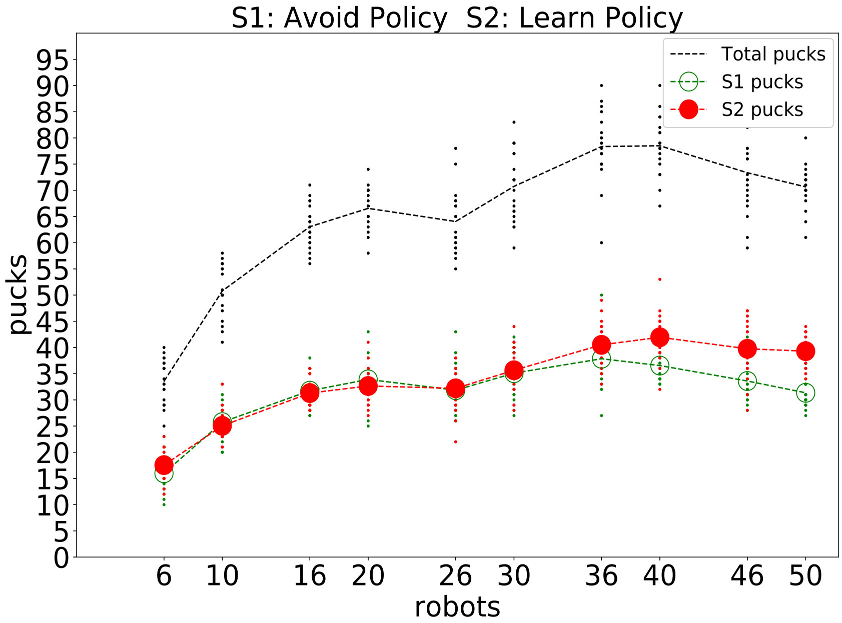
To examine the partition of the time invested between interaction and program execution (foraging), as noted by the robots, Fig. 6.3 shows the partition of the total time $\frac{T(a)}{T(a)+T(p)}$, as a function of the robot densities in the arena, for the experiment shown in Fig. 6.2. The blue dashed line represents the total percentage of interaction time, $\frac{T(a)}{T(a)+T(p)}$, for swarm S_2, which uses *attack* policy. The red line represents the total percentage of interaction time, for swarm S_1, which uses *avoid* policy. It is evident that S_1 spends less time in interaction.

For swarm S_2, the green and black lines represent the percentage of inter-swarm and intra-swarm interaction times, respectively. Similarly, for swarm S_1, the magenta and cyan lines represent the percentage of inter-swarm and intra-swarm interaction times, respectively.

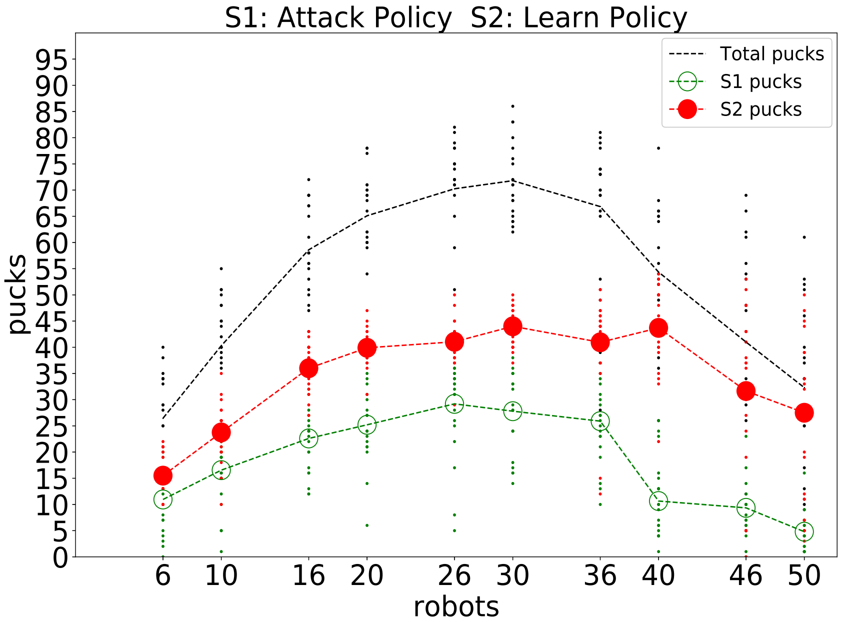
It is evident that the swarm used the *attack* policy, invests larger portion of its time in interacting rather than collecting pucks. Most of its interaction time is devoted to the other swarm rather than its own. This may justify that the local reward calculation by a robot during the learning process, should take both $T(a)$ and $T(p)$ into account. Even though swarm S_2 invest less time to collect pucks, as seen in Fig. 6.3, it still collects more pucks in high densities, as shown in Fig. 6.2. Unlike in a cooperative game of previous works, in a competitive game, it may be beneficial to invest time in interfering the opponents, rather then executing the main task of collecting pucks, which is reflected in the proposed theory and the reward function. Thus it might be beneficial to reward differently inter- and intra-swarm interactions, as their different contribution to the swarm's performance.

6.2.2 Experiments with Inter-swarm Learning

We now turn to experiments which involve learning. In the following scenarios, swarm S_1 executes a fixed, predefined policy, and swarm S_2 learns a preferred policy for inter-swarm interactions. The learning period lasts 50 minutes, and the learned policy is used during the measured game period which lasts 10 minutes. The results shown in the figures are of the measured time only, after the learning process had been completed.



(a) Green swarm S_1 uses *avoid* policy, red swarm S_2 learns policy.



(b) Green swarm S_1 uses *attack* policy, red swarm S_2 learns policy.

Figure 6.4: Number of pucks collected by two competitive swarms in a 10 minutes limited game, with learning involved.

Figures 6.4a and 6.4b, show results of applying the proposed learning model, in time-limited runs. To better analyze the results, Table 6.1 shows the distribution of learned policies within the learning swarm, S_2, by the end of the learning process. Note that since there are only two policies to learn from in this scenario, as described in subsection 5.2, the percentage of the *avoid* policy is the complement of the shown percentage of the *attack* policy. The columns of Table 6.1 relate to the scenarios of Figures 6.4a and 6.4b respectively.

In Figure 6.4a, S_1 uses *Avoid* policy, and S_2 learns the best policy based on the proposed model. As seen in the first column of Table 6.1, relates to the results of Fig. 6.4a, and similar to the results in Figures 6.1a and 6.1c, the learning process converges distinctly to avoid collisions in low densities, and converges distinctly to *attack* policy in in high densities, resulted with homogeneous swarm. In intermediate densities, the learning swarm is heterogeneous at the end of the learning process. As can be seen, the learning swarm, is always equal or better in scores.

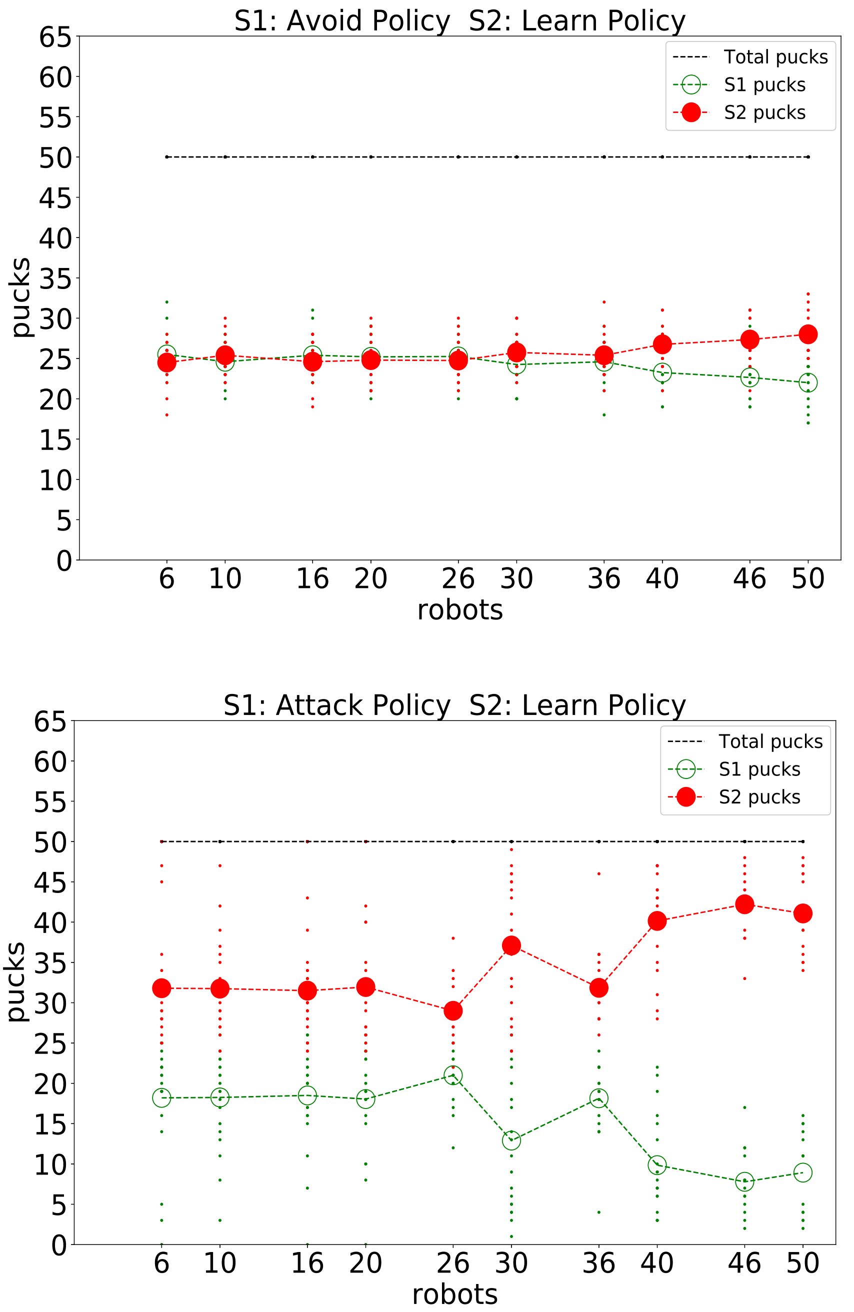
Number of players in game	Players learned to Attack in the experiment of Fig. (6.4a)	Players learned to Attack in the experiment of Fig. (6.4b)
6	1.6%	1.6%
10	2%	0%
16	2.5%	1.2%
20	7.5%	0%
26	31.9%	3%
30	61%	7.3%
36	97.7%	26.3%
40	99.7%	55.5%
46	100%	65.2%
50	100%	75.6%

Table 6.1: Percentage of players from swarm S_2 learned to use *Attack* policy.

In Figure 6.4b, S_1 uses *Attack* policy, and S_2 learns individually the best policy based on the proposed model. It is clearly seen from the figure that the learning swarm consistently outperforms the other swarm for any given density. At first it might be surprising that in high densities the score gap is so significant, based on fact that if both attack (Figure 6.1b), both swarms score low, and if one swarm use attack policy and the other use avoid policy, the attacking swarm scores better (Figure 6.1c).

The explanation to the result can be found the second column of Table 6.1. The learning swarm becomes heterogeneous under those conditions, since the robots of S_1 are occupied in interfering some robots in S_2, which mainly learn to employ *Attack* policy, due to high interaction rate and density. Thus, make room to remaining S_2 robots, which therefore sense a low-density environment, learn to use *Avoid* policy, and collect pucks in the remaining, more vacant area. Here again as in Fig. 6.4, the results support the use of the proposed learning model.

The two figure in 6.5 show the results of the two scenarios described in Fig. 6.4, but are puck-limited rather than time-limited.



l

(b) S_1 uses *Attack* policy, S_2 learned individual best policy using the proposed model. Score results are always win for S_2.

Figure 6.5: Number of pucks collected by two competitive swarms in a 50-puck limited game.

To conclude, first, learning swarms always performs equally or better than non-learning swarms (Figures 6.4a, 6.4b, 6.5a and 6.5b). Second, competitive behavior increases the total collected pucks, cumulative of all swarms (Figure 6.1). Third, after a learning process, a swarm may be either homogeneous or heterogeneous. Furthermore, in low-density environments, a robot better concentrate on its major goal, to collect pucks (*Avoid* policy); unlike in high density environments, where some robots better concentrate in interfering the other swarm (*Attack* policy), and potentially make room for its own swarm's other members to collect pucks (Figure 6.1c). The last point shows the strength of the proposed model. This is mainly since although a robot which is occupied in attacking other robots, cannot directly fulfill the goal of collecting pucks, it still optimizes the overall performance of its own swarm to score better that the other swarms.

Since the results in both limiting conditions, of time and puck limit, provide the same insights, from this point forward we show the results of time-limited runs only.

6.2.3 Adding Intra-swarm Learning

We now show that the learning process works also within the swarm as well, i.e., intra-swarm interactions. We performed intra-swarm learning versus a non-learning swarm, as shown in Figures 6.7, 6.6 and 6.8. The expectation is to see a convergence of the learning robots to avoid collisions with robots from their own swarm, at every density tested. The learning period lasts 50 minutes, and the learned policy is used during the measured game period which lasts 10 minutes.

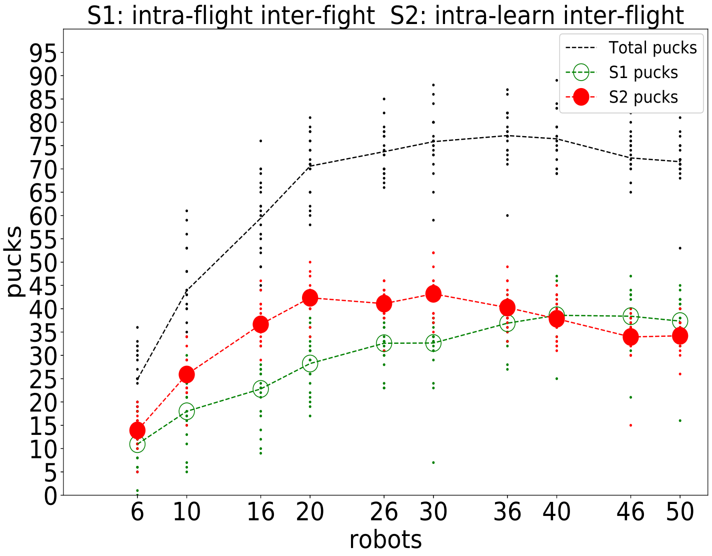


Figure 6.6: Number of pucks collected by two competitive swarms, where swarm S_2 Learns in intra-swarm and Avoids in inter-swarm, while swarm S_1 Attack in inter-swarm.

In Fig. 6.6, swarm S_2 learns how to act in intra-swarm collisions, and avoids inter-swarm collisions. It competes swarm S_1, a non-learning swarm, which always avoids collisions inside its swarm, and attacks in inter-swarm collisions.

In Fig. 6.7 the learning swarm, S_2, learns how to act with robots within its swarm (intra-swarm collisions), and always avoids collisions with robots from the other swarm (inter-swarm collisions). It competes with a non-learning swarm, S_1, which always avoids collisions, both inter- and intra-swarm.

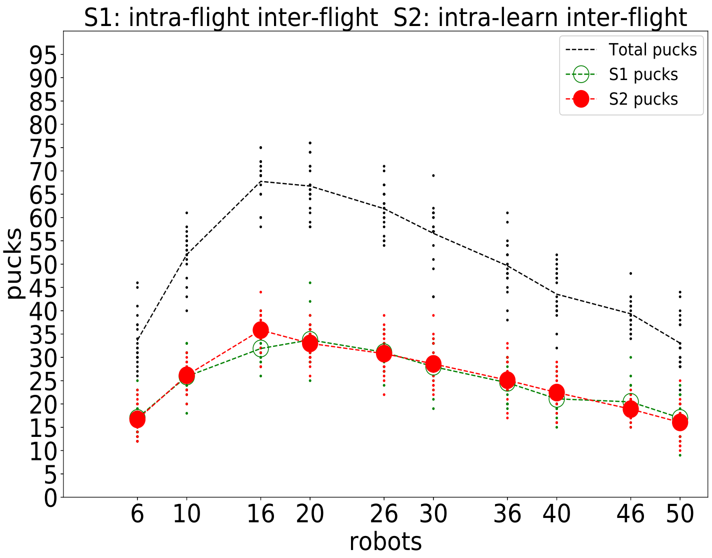


Figure 6.7: Number of pucks collected by two competitive swarms, where swarm S_2 Learns in intra-swarm and Avoids in inter-swarm, while swarm S_1 Avoids in all cases.

In both experiments, shown in Figures 6.7 and 6.6, more than 99% of all robots learned to avoid collision within the swarm. Thus, the results shown in these figures, are similar to the results of the corresponding experiments, shown in Figures 6.1c and 6.1a, where the intra-swarm policy was fixed to be *avoid*.

In the last figure, Fig. 6.8, the robots used double learning, both for inter- and intra-swarm collisions. Note that the two learning processes are completely separated, thanks to the fact that robots can distinguish the type of robot next to them. The learning swarm, S_2, competes a non-learning swarm, S_1, that always avoids collisions. As can be seen, the learning swarm, is always equal or better in scores.

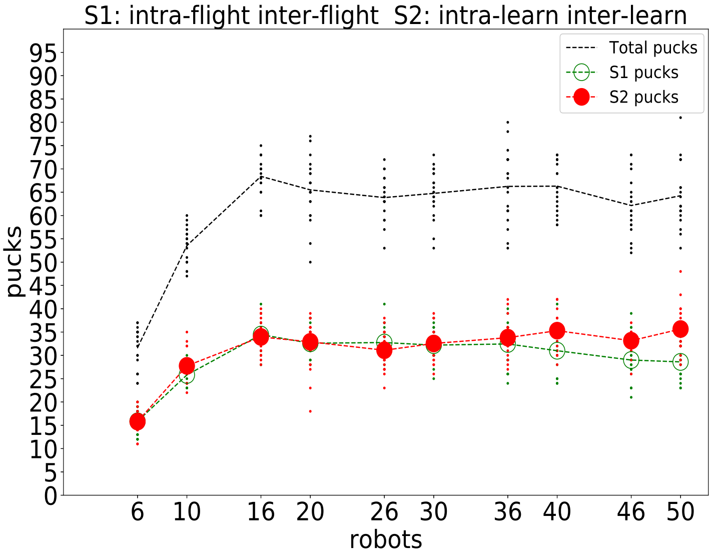


Figure 6.8: Number of pucks collected by two competitive swarms, where swarm S_2 learns both for inter- and intra-swarm, while S_1 avoids collisions.

Table 6.2 shows the distribution of the actions chosen at the end of the learning process, for the experiment shown in Fig. 6.8. The table shows percentage of learned policies for players from the learning swarm, S_2, at the the end of the learning period, as a function of the player density in the arena. The right column shows the percentage of players learned to avoid intra-swarm collisions. As seen, more than 99% of all robots learned to avoid collision within the swarm, regardless of the player density. The middle column shows the percentage of players learned to attack in inter-swarm collisions. Its complement to 100% chose to avoid for inter-swarm collisions, and is strongly depends on the density, which matches inter-swarm learning in the previous subsection.

Number of players in game	Percentage of players learned to Attack in inter-swarm collisions	Percentage of players learned to Avoid intra-swarm collisions
6	0%	100%
10	1%	100%
16	1.8%	100%
20	1.5%	100%
26	15.7%	100%
30	37.6%	100%
36	56.9%	99.7%
40	71.25%	99.7%
46	75.4%	99.8%
50	77.8%	99.4%

Table 6.2: Percentage of policies learned by players from swarm S_2 in the experiment of Fig. 6.8.

The results described in this subsection, show that intra-swarm interaction always choose the *flight* method, which matches the negative marginal reward associated with intra-swarm interactions.

To conclude this section, we examined baseline cases of predefined policies, and performed inter- and intra-swarm learning. We supported that investing more time in interaction may sometimes be more beneficial than program time in inter-swarm interaction, and as a side effect increased the total number obtained by all swarms.

6.3 Additional Cases

In the previous subsection we have simulated baseline scenarios, where a learning swarm was competing a non-learning swarm under the proposed model, with no changes. In this subsection, we explore the performance of the learning swarm, where some changes to the above model were applied. More specifically, we explore changes in the approximated reward and selection methods used during the learning process (subsections 6.3.1, 6.3.2 and 6.3.4), changes in the experiments' environment (subsection 6.3.3). In addition we explore the case of learning in a single swarm (subsection 6.3.5).

6.3.1 The Use of WLU Based Reward

In order to justify our use of the WLU function, described and calculated in previous chapters, in player's rewards during the learning process, we also examined learning based solely on the individual utility function U_i. Since the WLU function of each player takes into account both its own utility and its impact on the other players, we expected differences in the learning process results, and therefore in the total reward at the end of the game. As shown later in this subsection, the swarm which performed learning that takes into account only its players' own utility without considering its impact on the environment, i.e. the swarm that studied without WLU function, got similar or lower results than those of the swarm learning using the WLU function. The individual utility function (non-WLU) based learning got even lower in some cases than the swarms performing fixed actions without learning. This experiment illustrates the benefit of adding the players' impact on their environment in addition to its own individual utility, namely the WLU.

In the case where player i in swarm k, i.e., $i \in S_k$, using Eq. (4.7), and the assumptions used in Section (5.3), the utility is:

$$\begin{aligned}
U_i = {} & u_i^k + u_i^m \\
= & n_k \cdot (\beta_i \cdot T(p) - \alpha_i \cdot T(a)) + n_m \cdot \big((\beta_i \cdot T(p) - \alpha_i \cdot T(a)) - \big(\overline{\beta_m} \cdot T(p) - \overline{\alpha_m} \cdot T(a)\big)\big) \\
= & \big(n_k \cdot \beta_i + n_m \cdot \beta_i - n_m \cdot \overline{\beta_m}\big) \cdot T(p) + (n_m \cdot \overline{\alpha_m} - n_m \cdot \alpha_i - n_k \cdot \alpha_i) \cdot T(a)
\end{aligned}$$

As before, in this experiment we consider all players to have the same capabilities during their program execution, therefore their probability of finding a puck at a certain time is equal, and we can write $\beta_i = \beta_j = \beta$. As in for the reward function using WLU, Eq. 5.4, we can write:

$$\begin{aligned}
\frac{U_i}{n} = & \frac{n_k \cdot \beta}{n} \cdot T(p_i) + \frac{(n_m \cdot (\overline{\alpha_j} - \alpha_i) - n_k \cdot \alpha_i)}{n} \cdot T(a_i) \\
r_i \approx & \frac{n_k}{n} \cdot \beta \cdot T(p_i) - \frac{n_k}{n} \cdot \Delta(\alpha) \cdot T(a_i) \qquad (6.3)
\end{aligned}$$

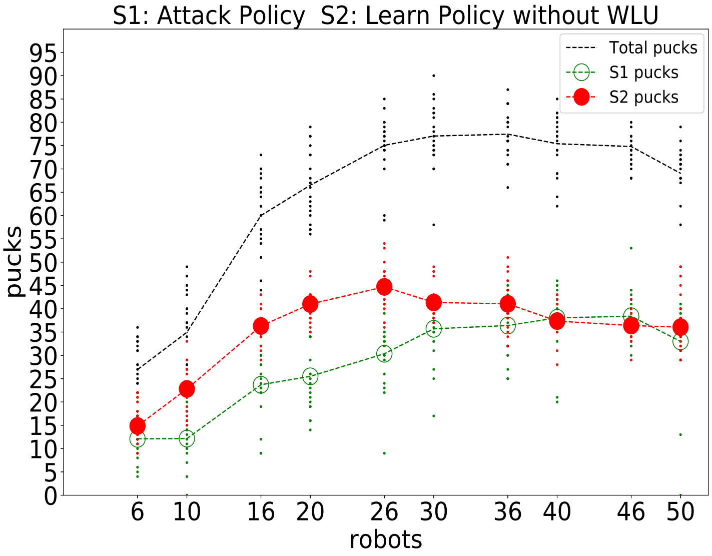
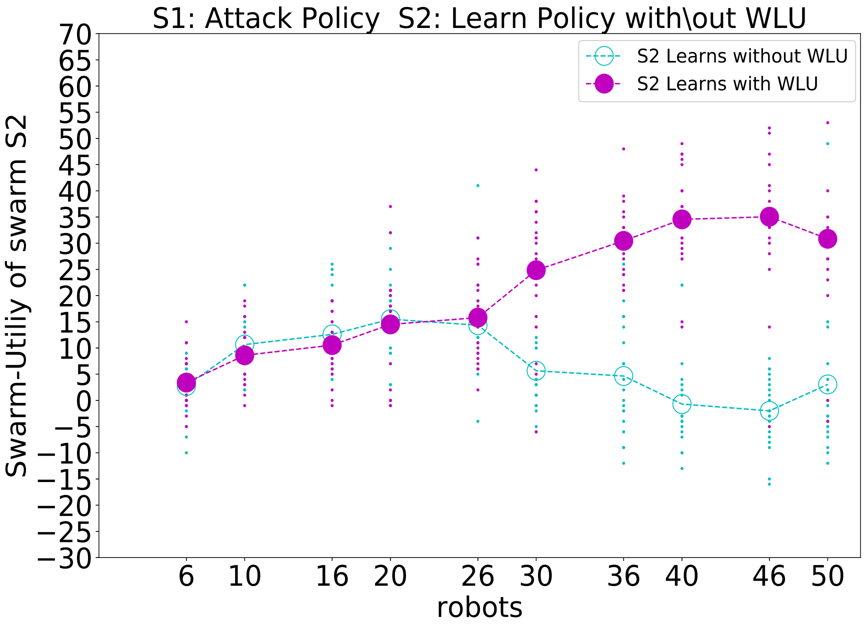


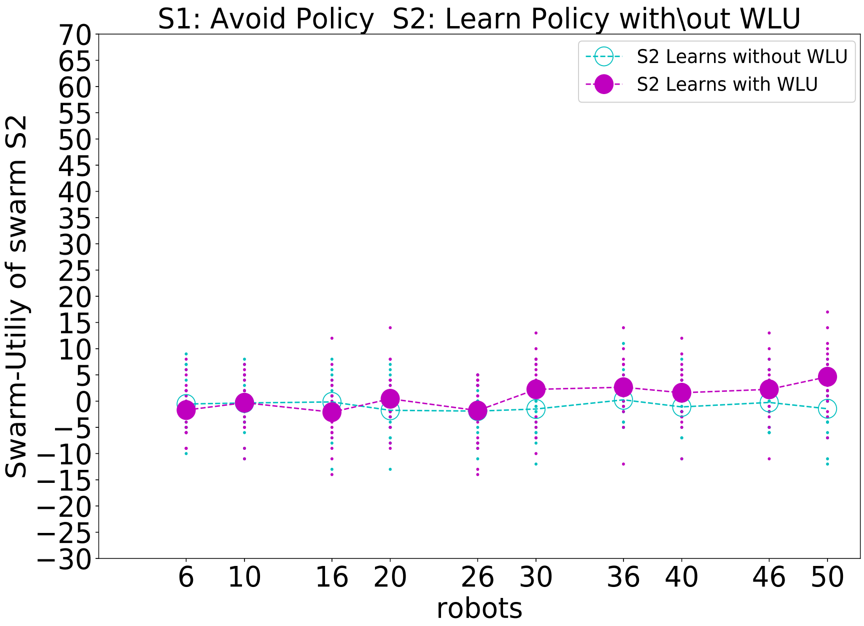
Figure 6.9: Pucks collected by non-WLU based learning swarm S_2, and attack non-learning swarm S_1.

Fig. 6.9 shows the results achieved by two competing swarms, where the red swarm, learned using individual utility (non-WLU) as the reward, using Eq. (6.3), and the green opponent swarm has attacking players. It is evident that the learning process still contributes to better or equal results when compared to non-learners.

Fig. 6.10a compares the Swarm Utility Difference (SUD), which is the difference between the total utility got by the swarms, of two different experiments. The SUD of the experiment shown in Fig. 6.9 (namely the difference between the red and the green lines) is shown in Fig. 6.10a in cyan. The SUD marked in magenta dashed-line in Figure 6.10a is the SUD got by a WLU-based learning swarm from the experiment shown in Fig. 6.5b. Comparing the results in Fig. 6.10a demonstrates the advantage of learning using WLU function rather than ignoring the influence of the individual player on its environment. In other words, the magenta WLU-based SUD is always equal or higher than the cyan non-WLU-based SUD, which justifies WLU-based learning.



(a) The SUD of two different learning-swarms got by playing versus an enemy-attackers swarm.



(b) The SUD of two different learning-swarms got by playing versus a swarm that avoids collisions.

Figure 6.10: A comparison of the Swarm Utility Difference (SUD), $\mathbb{U}_{S_2}$, of different learning-swarms S_2 competing with a swarm of a non-learning swarm S_1. The SUD of the swarm that learns non-WLU-based reward is marked in cyan and the swarm learns with WLU-based reward is marked in magenta. WLU-based reward is always equal or better.

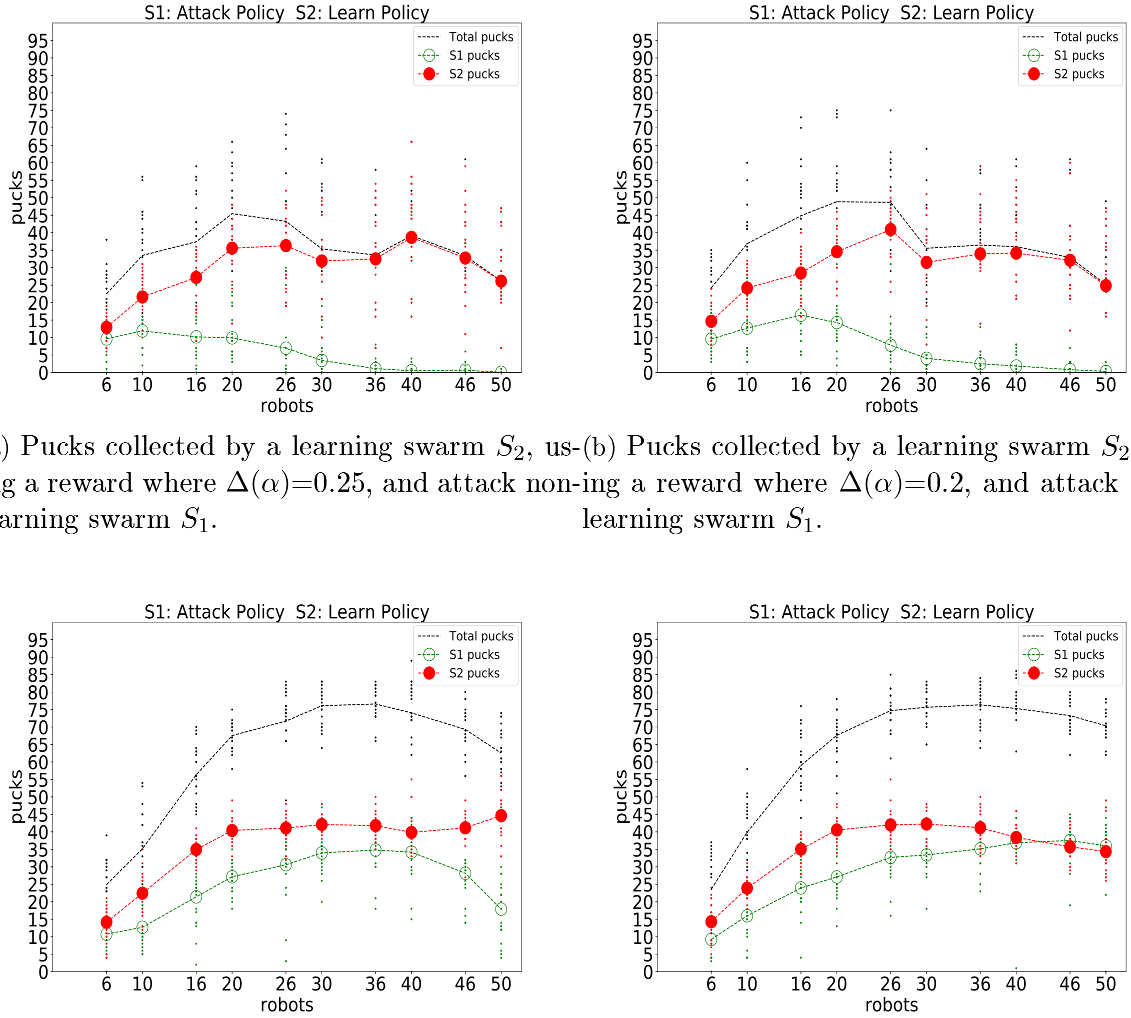
A comparison of the non-WLU learning and WLU-based learning SUD is shown in Fig. 6.10b, where in these cases the learning swarms competes a swarm that uses avoid policy. The magenta dashed line, marked with filled circles, is for the WLU-based learning swarm, and cyan dashed line, marked with unfilled circles, is for the non-WLU learning swarm. In this case, where the original gain of learning was small, the results with non-WLU function were still similar. As a reminder, since we deal with zero-sum games, the other SUD, $\mathbb{U}_{S_m}$ is not shown in the figure, but is known to be the negative of $\mathbb{U}_{S_k}$.

To summarize, in this subsection we demonstrated that learning base on WLU-function performs equally or better than learning based of individual utility. In the next subsection we explore the approximation of the reward, based on the proposed model's WLU function.

6.3.2 Rewards Comparison

In this subsection we explore different approximation parameters for the calculation of the individual robot's reward. So far, in this chapter we assume identical behavior during foraging program execution time, and therefore the contribution of program time is identical for all robots, regardless of their swarm identities. We do not assume identical contribution during the interaction time, therefore we explore the influence of the different α values for different players.

Since a robot cannot be aware of the contribution of collision time to its own and to its surroundings rewards, we used an approximation in Eq. (5.4). To verify the influence of the specific value of $\Delta(\alpha)$ we tested a variety of values and examined their effect on game results, i.e. on the number of pucks collected during the game. In particular, we examined the effect of the α, the multiplier of $T(a)$, values on the SUD, which is the difference between the number of pucks collected by the learning swarm S_2 versus the number of pucks collected by the non-learning swarm S_1. We concentrated on the attack policy swarm, as the opponent, where the resulted swarm utility values were significantly better for learning swarms.



(a) Pucks collected by a learning swarm S_2, using a reward where $\Delta(\alpha)$=0.25, and attack non-learning swarm S_1.

(b) Pucks collected by a learning swarm S_2, using a reward where $\Delta(\alpha)$=0.2, and attack non-learning swarm S_1.

(c) Pucks collected by a learning swarm S_2, using a reward where $\Delta(\alpha)$=0.08, and attack non-learning swarm S_1.

(d) Pucks collected by a learning swarm S_2, using a reward where $\Delta(\alpha)$=0.02, and attack non-learning swarm S_1.

Figure 6.11: Number of pucks collected by two competitive swarms in a 10 minutes limited game, after 50 minutes of learning, using different learning rewards.

Figure 6.11 shows the number of pucks collected versus robot densities, for competitions between a non-learning attacking swarm (green), and a learning swarm (red). The learning swarm used the approximation of Eq. (5.4) with the parameter $\Delta(\alpha)$ of 0.25, 0.2, 0.08 and 0.02, instead of 0.125 used in other sections and in the baseline cases. It is evident that changing this parameter, influences the SUD results. Therefore, we compared a range of $\Delta(\alpha)$ values in Fig. 6.12.

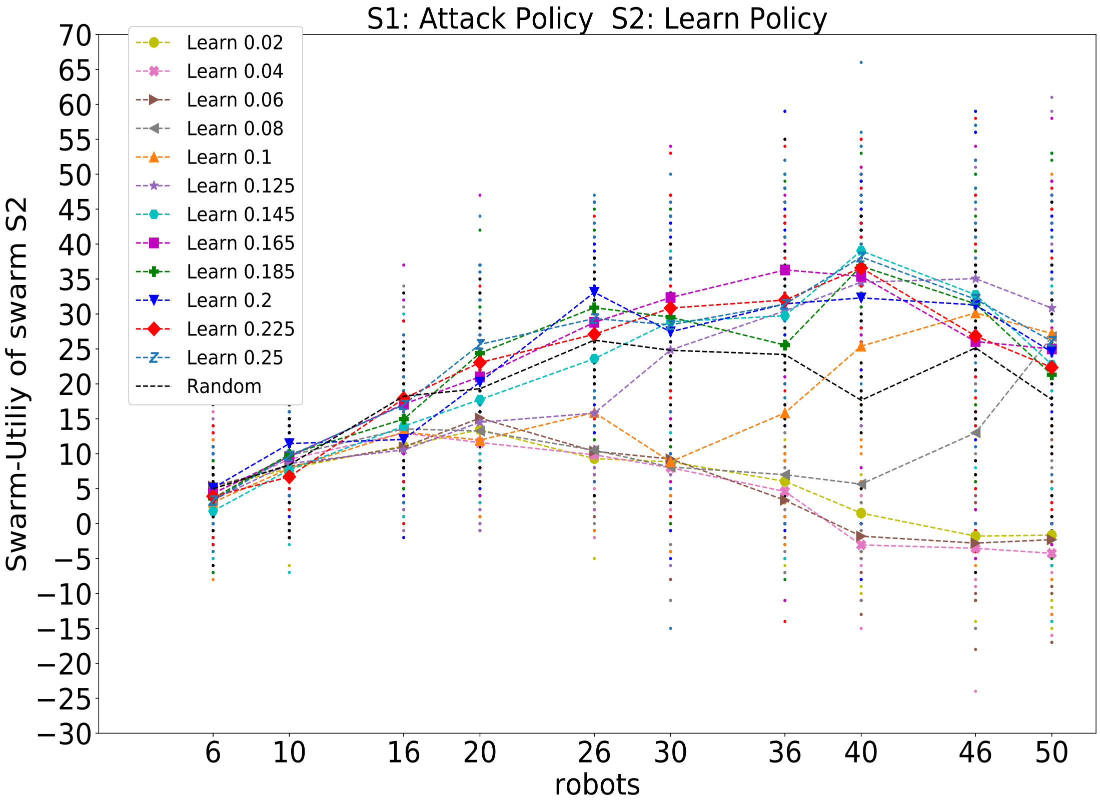


Figure 6.12: The SUD $\mathbb{U}_{S_2}$, of learning-swarm S_2 using different $\Delta(\alpha)$ values in reward, the plays versus faster swarm S_1, that attack all players in swarm S_2.

Figure 6.12 shows a comparison of the swarm-utilities of the learning swarm when competing a non-learning, attacking swarm, in different robot densities in the arena. Each dashed line in the figure marks the SUD achieved using a different $\Delta(\alpha)$ value in the reward of Eq. 6.1. Where the values were ranging between 0.02 to 0.25. It is evident from the figure that as the robots density in the arena increases, the spread of the resulted SUD also spreads. For low densities, all tested cases gave similar results. One can explain that the learning process affects behavior during conflict resolution times, and since there are fewer collisions in lower densities, the differences are minor. Low values of $\Delta(\alpha)$ provide some lower bound on the possible performance, which leads to very poor performance at high densities, resulted swarm-utilities of zero or negative. Upper $\Delta(\alpha)$ tend to form an upper bound on performance, with different values taking the maximum results for specific robot densities. It is interesting to say the the random action selection gives performance between the lower bound and the upper bound for all densities. Another observation that is clearly seen in the figure, is that as $\Delta(\alpha)$ increases, the separation from the lower bound occurs at lower densities.

In most of this work, we have used the value $\Delta(\alpha) = 0.125$, due to a number of reasons. First, for highest densities this parameter provides the best SUD values, under the defined environment. One should bare in mind that the figure shows results of a specific opponent swarm type, total of two swarms in the arena, and specific arena. Since this work examines various cases with different hyper parameters used in this subsection, this value was selected to avoid over-fitting. Future work may explore the use of different $\Delta(\alpha)$ values as a function of the robots densities and other environmental parameters.

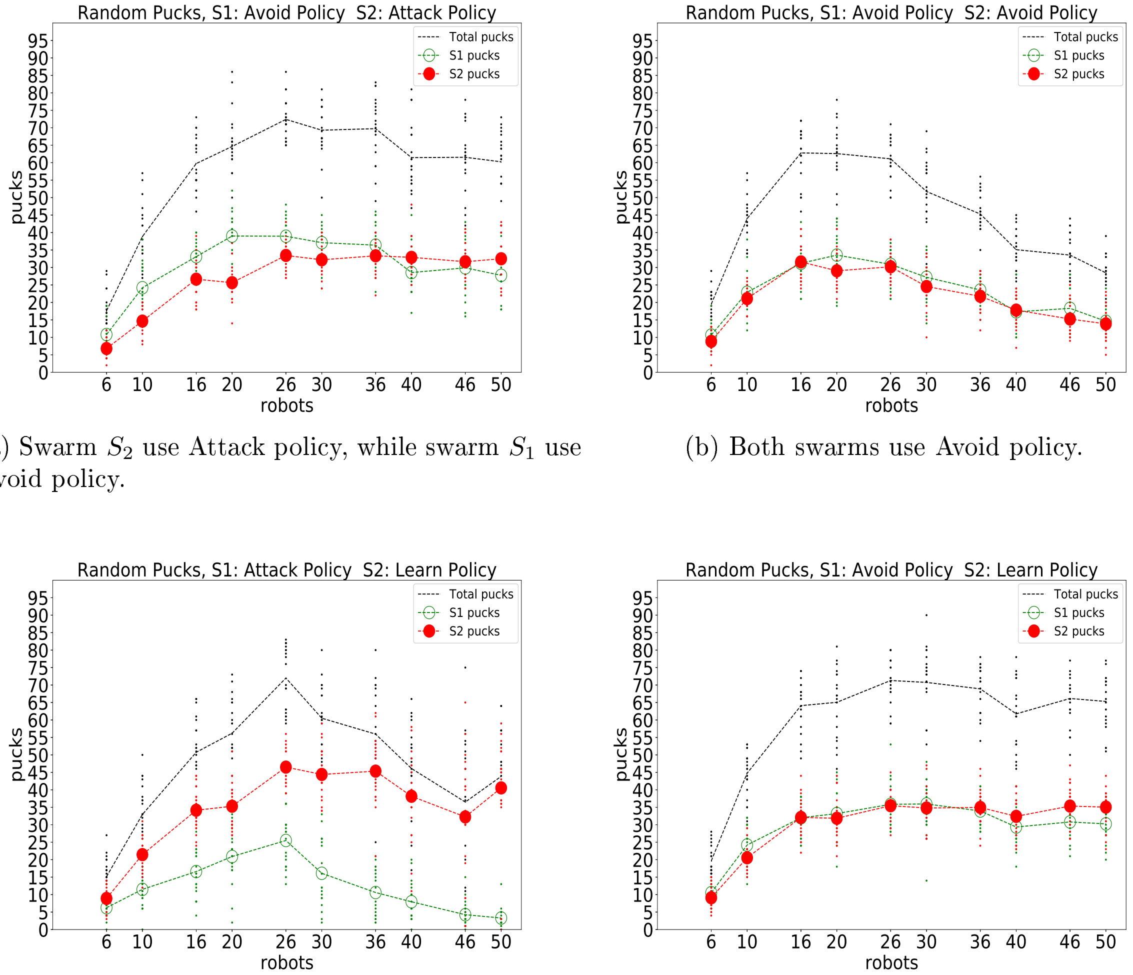
Number of players in game	Percent of players learned to Attack with reward $R_{0.02}$	Percent of players learned to Attack with reward $R_{0.08}$	Percent of players learned to Attack with reward $R_{0.125}$	Percent of players learned to Attack with reward $R_{0.2}$	Percent of players learned to Attack with reward $R_{0.25}$
6	0%	0%	1.6%	1.6%	8.3%
10	0%	0%	1%	12%	18%
16	0%	0%	2.5%	26.8%	29.3%
20	0%	0.5%	2%	34.5%	35.5%
26	0%	0.8%	11.5%	46.9%	47.6%
30	0%	1.6%	33.3%	57.3%	57.6%
36	0%	2.2%	51.1%	62.7%	65%
40	0%	4.25%	49.25%	63.5%	64%
46	0%	17.6%	58.2%	69.1%	72.6%
50	0%	32.6%	56.4%	76.8%	78.4%

Table 6.3: Percentage of players from swarm S_2 learned to use Attack policy in the experiment of Fig. (6.12).

Table 6.3 shows the percent of robots chose the *Attack* policy rather than *Avoid* policy at the end of the learning process. One can observe that in this tested scenario, the percentage of players who choose to *Attack* is larger as the reward parameter is larger, but this percentage is blocked at a certain density. The distribution of the players' decisions affects the total number of pucks collected by the entire swarm as can be seen in the Fig. 6.12. It makes sense that the difference between the $\Delta(\alpha)$ parameters of the reward affects the degree of aggressiveness of the players, since the α parameter represents the contribution of the collision time on the player's utility. In other words, a player who benefits from longer interaction learns to attack more, and a player who benefits from concentrating on the game rather than colliding learn to avoid collisions to maximize its program execution time.

6.3.3 Robustness to Initial Conditions

As described in Section 5.1, the standard experiments assumed fixed locations of home bases and pucks, and were symmetric in location. However, since initial position of the robots was random, there could still be an unintentional advantage to one of the swarms. Therefore, the following experiments tested 200 runs in 200 different initial puck locations, which resulted with behavior similar to previous experiments. Figures 6.13a, 6.13b, 6.13c and 6.13d should be contrasted with Figures 6.1c, 6.1a, 6.4b and 6.4a respectively.



(a) Swarm S_2 use Attack policy, while swarm S_1 use Avoid policy.

(b) Both swarms use Avoid policy.

(c) Swarm S_2 Learns policy, while swarm S_1 use Attack policy.

(d) Swarm S_2 Learns policy, while swarm S_1 use Avoid policy.

Figure 6.13: Number of pucks collected by two competitive swarms in a 10 minutes limited game, after 50 minutes of learning.

For the non-learning scenarios, figures 6.13a and 6.13b, we achieved the exact closely matching results to our standard experiment configuration. In the other two scenarios there were minor differences, which still kept the same general behavior we saw in the standard experiments. However, those relatively small differences, gave the idea to also check what happens when one of the swarms has intentionally some kind of advantage over the other. In subsection 7.5 we locate the pucks closer to the non-learning swarm and check if the learning swarm could overcome this disadvantage.

To conclude, the results of examining the learning process in different environments, were independent of initial puck location. We can therefore assume, that the standard experiment configuration is general enough and the learning process behaves similarly both in symmetric and asymmetric initial puck location.

6.3.4 Comparing Different Exploration and Selection Methods

As discussed in previous chapters, during learning, a player selects actions to perform based on a select method, then performs the selected action, then measures the reward, and updates the estimated values of the action. Furthermore, in this work we use Upper Confidence Bound (UCB) method as a selection method; Multi-Armed Bandit (MAB) learning algorithm; and the reward based on zero-sum game, player's utility and the mutual influence of the player on the other players and vice versa, using the Wonderful Life Utility (WLU) function. In this subsection, we replaced the UCB with alternative methods detailed below, to demonstrate the generality of use of the rest of the model. We therefore experimented the same scenario, where the difference for the learning swarm was from the following alternatives: Discounted UCB (DUCB) described in [21], a greedy method, and a periodic random exploration using two different parameters. All learning process are used with the same WLU-based reward function.

The comparison shows that our WLU-based reward function is applicable not only to the UCB selection method, but also to other known selection methods.

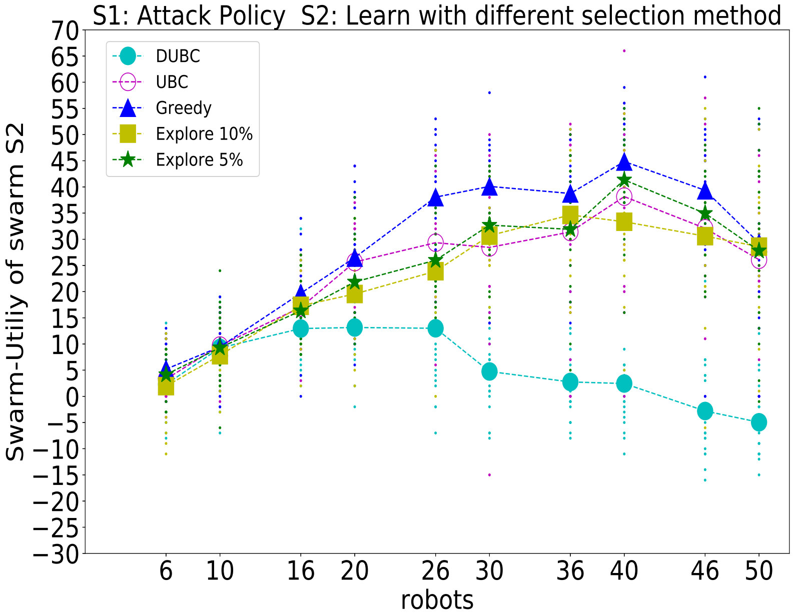


Figure 6.14: The Swarm Utility Difference (SUD) $\mathbb{U}_{S_2}$, of learning-swarm S_2 using different selection methods during the learning process. The reward function is equal for all learning swarms.

Figure 6.14 compares the SUD values $\mathbb{U}_{S_2}$, which is the difference between the number of pucks collected by each swarm, when compete with a non-learning, attacking swarm. Each curve shows the SUD for a set of runs, where the learning swarm used different method selection, while the calculation, learning reward and action's value estimation are the identical to the standard scenario. It is evident from the figure that most of these alternative methods, as the one specifically proposed earlier (UCB based), provides significantly positive SUD values, mainly in the higher densities. The results are surprising in two ways. First, the greedy method outperforms UCB. Second, the Discounted UCB, which is intended for distributed learning settings, is outperformed by all methods. One possible explanation for the DUCB results is that the WLU-based reward function is already considering the distributed learning environment, and the impact of the player's actions to its surrounding.

6.3.5 Learning in a Single Swarm

This subsection deals with the single swarm scenario, and tries to assess the learning ability in the single-swarm case, thus be a generalization of earlier results. In this case, four different possible actions from two possible strategies were experimented. In these experiments, the pucks were collected by a single swarm in a standard time-limited, in a standard 10-minutes game. There were five scenarios, four of fixed policies where no learning was performed, and all robots always chose the same predefined action during interaction, and one scenario where each robot learned a preferred action according to the proposed model. In the four fixed actions, three actions are of *flight* type, as described in subsection 5.2. As a reminder, in this action the robot finds the most vacant direction (less robots detected), rotates towards this direction and drives forward for a predetermined time. In this experiments the time durations were chosen to be 50, 500 and 1000 ms. In addition, a forth action that was allowed to the robots to learn was a *fight* action, in which the robot drives forward for a fixed duration of 1000 ms towards the robot which it interacts with. In the fifth scenario, robots in the swarm learned to choose one of the above four actions.

Adapting the utility of a single player to the single swarm case based on the proposed theory, the reward used by the robots during learning, using Eq. (4.2), (4.10) and (4.13):

$$\begin{aligned} WLU_i &= u_i^k + \phi^i_{S_k} \\ &= n_k \cdot (\beta \cdot T(p) - \alpha_i \cdot T(a)) + \epsilon^{\rho}_{S_k} - \epsilon^{\overline{\rho}}_{S_k} \\ &= n_k \cdot (\beta \cdot T(p) - \alpha_i \cdot T(a)) + n_k \cdot (\beta \cdot T(p) - \overline{\alpha_k} \cdot T(a)) - n_k \cdot \beta \cdot (T(p) + T(a)) \\ &= n_k \cdot (\beta \cdot T(p) - (\alpha_i + \overline{\alpha_k} + \beta) \cdot T(a)) \end{aligned}$$

Figure 6.15 shows the number of pucks collected by the swarm. Note that the single swarm case, is not a zero-sum game, and the SUD becomes the number of pucks collected. It can be seen from the figure that the different actions perform differently. The worse action of fight (black) results from the swarm being occupied with interactions. The cyan, magenta and yellow lines are for the three flight methods. The learning swarm achieved exactly the highest possible score when compered to all other actions (the two curves, yellow of flight with duration of 50 ms, and blue of learn are overlapping). The swarm has learned almost universally to use the most effective action for a game without opponents, which is the action that avoids collision in the fastest time, and thus, the robots can return to collect pucks.

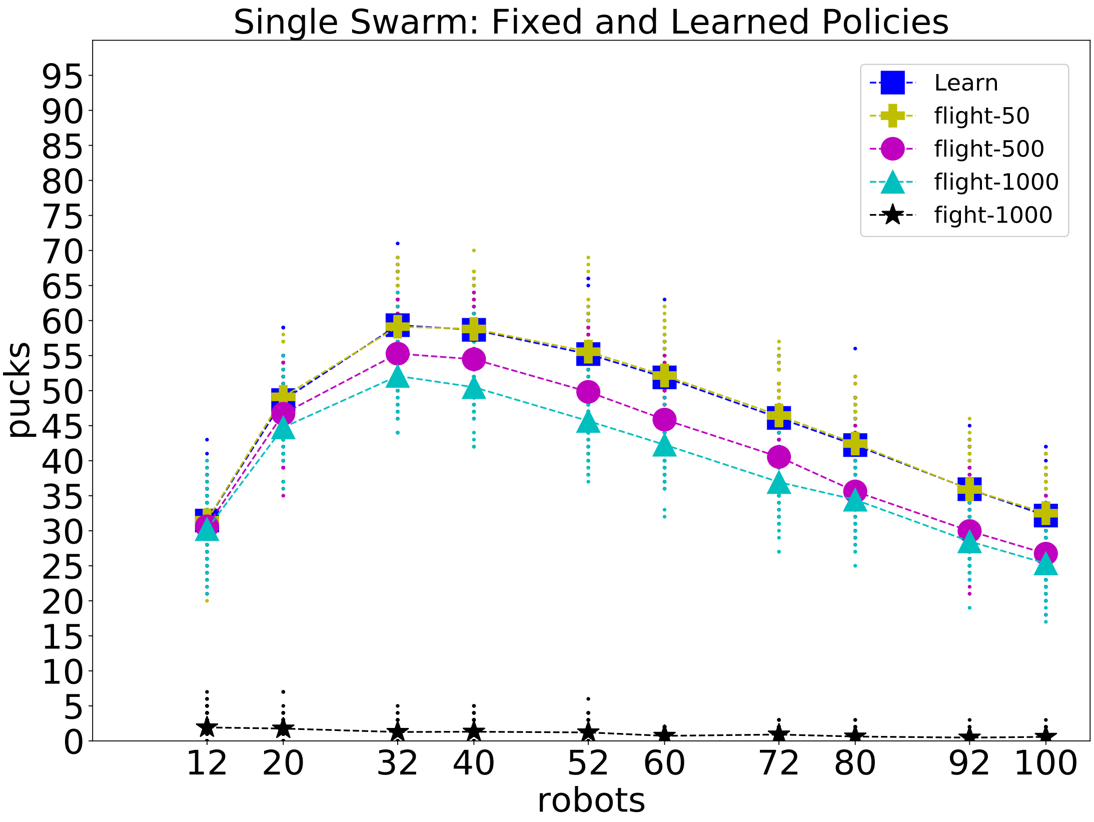


Figure 6.15: Number of pucks collected by a single swarm using different collision actions and during intra-swarm learning. The learned result (yellow) overlaps the best non-learned results (blue).

The experiment shows that the learning method which is suitable for two swarms is also suitable in the case of one and only swarm, and examines a case similar to the one tested in previous works of [18, 43].

Chapter 7

Asymmetric Competition

In the previous chapter we concentrated on the case of identical competing swarms, i.e, during the program execution state (not during conflict resolution), the robots behave exactly the same when searching for pucks and moving to their respective homebases. In this chapter, we now elaborate on the case of non-identical program executing behavior for different swarms, and in particular when one of the swarms has some kind of advantage over the other, and even a case of three swarms competition. We explore the case of competing a faster swarm, an uneven number of robots in the swarms, and a case of advantaged location of homebase in the arena. When in disadvantage, being equal to the opponent is insufficient to win, and the learning swarm needs to overcome the opponent's advantage gap.

Section 7.1 applies the asymmetric conditions to reward function, for the following sections of this chapter. Section 7.2 analyze the case where the learning swarm is slower than the other during the program execution time. Section 7.3 demonstrates the case of three-swarm competition, where the learning swarm is slower than the others. Section 7.4 shows the case where the learning swarm is disadvantaged by having less players in the swarm than the other. Section 7.5 shows a case of a learning swarm with disadvantaged location of homebase in the arena.

7.1 The Reward Function for Asymmetric Swarm

As a reminder, Chapter 4 defined the parameters α and β as the contribution of the interaction and program execution times, respectively to the utility. In the previous chapter, we assumed that there is no inherent advantage to any of the swarms or robots, therefore, their β parameters were equal. In this chapter, we give the non-learning swarm an advantage, and therefore its chance to get more pucks in a given time interval is higher.

We now relate to the case where the non-learning swarm has an advantage over the learning swarm, which is demonstrated in the rest of this chapter. The advantage is related to the program execution, foraging, rather than during interactions. Such advantage could be created through higher speed, better homebase location, bigger swarm, etc. Therefore, it would probably influence the β parameters rather than the α parameters. In all experiments we kept the advantage ratio to be $\frac{7}{6}$ as detailed in the next sections. Our goal now is to reflects this disadvantage of the learning swarm into the learning process by affecting the reward function parameters.

We estimate the reward for the scenario where the program execution is not identical, i.e $\beta_i \neq \beta_j$. Since the advantage ratio of the swarms is $\frac{7}{6}$, we assign

$$\beta_j = \overline{\beta_m} = \frac{7}{6} \cdot \beta_i = \frac{7}{6} \cdot \overline{\beta_k}$$

into Eq. (5.2), to get the WLU of player i, in order to estimate the reward:

$$\begin{aligned}\frac{W_i^{\mathbb{U}s_k}}{n} = &\left(\left(\frac{n_k + \frac{5}{6} \cdot n_m}{n}\right) \cdot \beta_i\right) \cdot T(p) + \left(\left(\frac{\frac{7}{6} \cdot n_m - n_k}{n}\right) \cdot \beta_i\right) \cdot T(a) \\ &+ \left(\frac{2 \cdot n_m \cdot \overline{\alpha_m} - n_k \cdot \alpha_i - n_k \cdot \overline{\alpha_k} - 2 \cdot n_m \cdot \alpha_i}{n}\right) \cdot T(a) \end{aligned} \tag{7.1}$$

Therefore we can rewrite Eq. (5.4), using Eq. (7.1), to get the approximated reward for the disadvantaged learning swarm case:

$$R = \frac{W_i^{\mathbb{U}_{S_k}}}{n} \approx \frac{11}{12} \cdot \beta \cdot T(p) + \left(\frac{n'_i}{n'} \cdot \sigma_i \cdot |\Delta(\alpha, \beta)| \right) \cdot T(a) \tag{7.2}$$

We use this reward in the learning process of the experiments detailed in the next sections, where different values are assigned to $\Delta(\alpha, \beta)$, and the effect on the results is demonstrated and discussed.

7.2 Competing a Faster Swarm

In this subsection we analyze the case where the robots on the arena are not homogeneous during the program execution. We assume the scenario where the robots in swarm S_1 are faster than the robots in swarm S_2. In other words, swarm S_2 has initial disadvantage when competing swarm S_1.

We first illustrate the advantage of S_1 with non-learning S_2. We then follow with faster swarm S_1 and learning swarm S_2 with a goal to defeat S_1. In this section we gave swarm S_1 speed of 70 , and swarm S_2 the speed of 60 cm/m.

7.2.1 Baseline: Two Non-learning Swarms with Speed Difference

In order to explore the benefit of learning, we would like to give the fixed behavior swarm an advantage. In the baseline one swarm has inherent greater speed during the program execution time, and therefore has an advantage over the other in collecting pucks. We started the sequence of experiments in this subsection with an experiment that verifies that swarm S_1 indeed has collected more pucks in the game since it is faster when this is the only difference between the swarms.

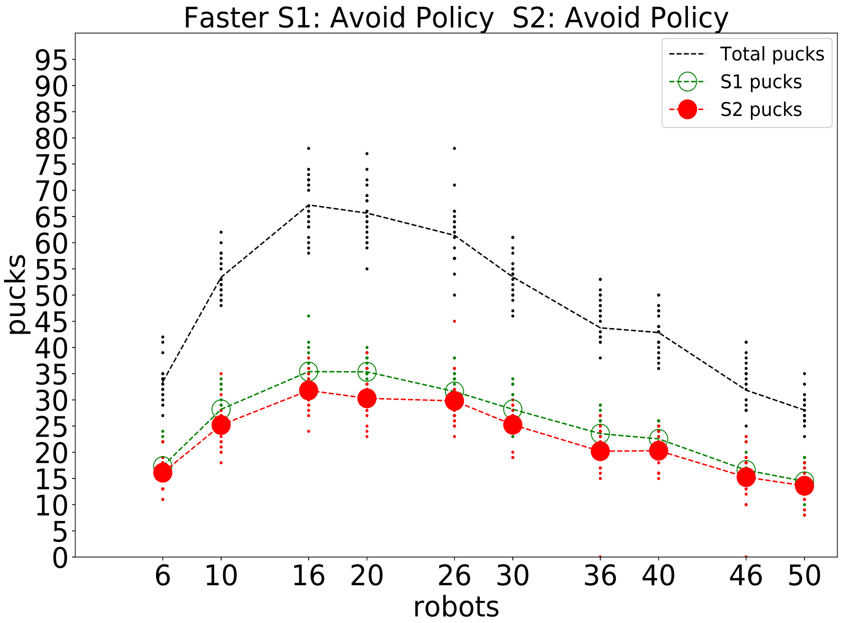
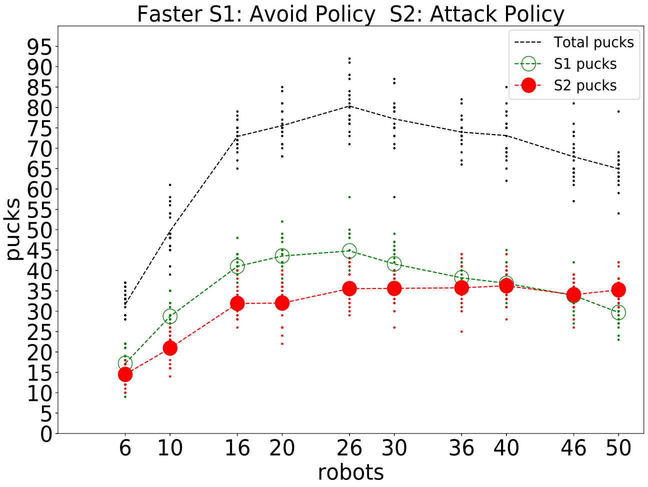


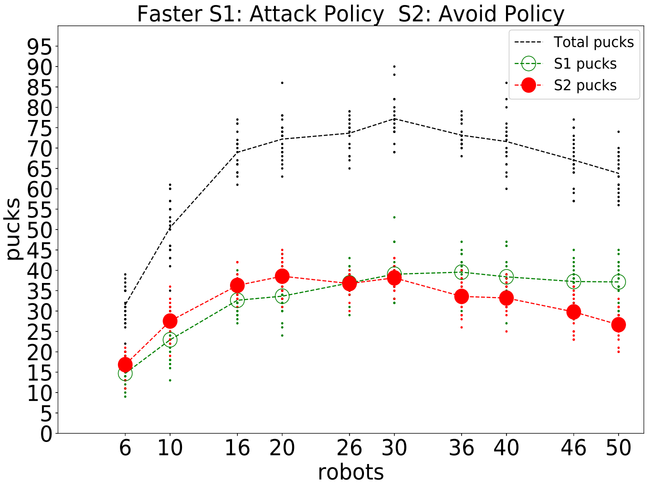
Figure 7.1: Swarm S_1 is faster than swarm S_2, both always avoid collisions. The advantage of swarm S_1 is consistent for all densities.

As shown in Figure 7.1, the faster swarm collects more pucks in every density. In order to compare the results of experiments that involve a learning process with different speeds, we ran two more experiments without learning. The experiment shown in Figure 7.2a, where swarm S_1 is faster than swarm S_2. The faster swarm S_1, uses *fight* method robots against the opponent swarm, i.e. attacks opponents (while avoiding collisions within its swarm). The slower swarm S_2, always uses *flight* method, i.e., avoids collisions. As shown in the figure, S_2 loses in almost all densities, get similar results in higher densities, and wins only in the most crowded game settings.

In the next experiment, shown in Figure 7.2b, swarm S_1, is still faster than swarm S_2, but the faster swarm S_1, uses *flight* method and the slower swarm S_2, uses *fight* method. As shown in the figure, S_2 wins in low densities, and loses in high densities.



(a) Pucks collected when faster swarm S_1 Avoids, slower swarm S_2 swarm Attacks.



(b) Pucks collected when faster swarm S_1 Attacks, slower swarm S_2 swarm Avoids.

Figure 7.2: Number of pucks collected by two competitive swarms in a 10 minutes limited game with different densities, where the swarms has different driving speeds of 60 and 70.

In the previous chapter, we saw that for identical swarms during program execution time, (Fig. 6.2) the flight method wins at lower densities, while the fight wins in higher densities, where the equal score was achieved for density of 36 robots on the arena (Fig. 6.1c). In the current runs, we see that the faster speed influences this "tie" point: S_1 wins from lower density till density of 46 when it uses flight method, (Fig 7.2) and S_1 wins from density 26.

In the next sections we apply learning in order to successfully overcome the learning-swarm's disadvantage. We test our learning model against four non-learning advantaged swarms.

7.2.2 Learning Swarm vs. a Faster Attacking-Swarm

We now apply learning in order to successfully overcome the learning-swarm's disadvantage of speed as described earlier in this section. The policy of the non-learning faster swarm is *Attack*, i.e, *fight* in inter-swarm collisions and *flight* in intra-swarm collisions. Fig. 7.3 provides the result of one of the examples mentioned in subsection 7.1, using reward of $R_{0.145}$, where $\Delta(\alpha, \beta) = 0.145$ in Eq. (7.2).

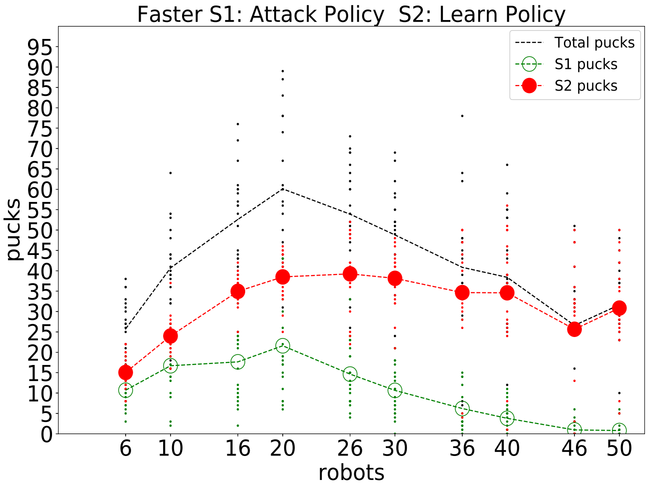


Figure 7.3: Swarm S_1 moves in 70 speed always attacks enemies, swarm S_2 move in 60 speed and learns how to act in collisions, using reward $R_{0.145}$.

Figure 7.4 compares the three examples of rewards mentioned in Subsection 7.1. As usual in this work, the x-axis represents the total number of robots on the field both swarms, i.e. the density. The y-axis represents the SUD received by the learning swarm, as defined in definition 4, i.e., the number of pucks collected by swarm S_2 minus the number of pucks collected by S_1. In order to simplify the comparison between the different rewards, we compare the swarm utilities, rather than the pucks collected for each swarm. The SUD of the experiment shown in Fig. 7.3 is represented by the yellow line in Fig. 7.4.

Each graph in the figures shows the results of an experiment run with a different reward function, from the rewards suggested above. The cyan graph represents the learning outcomes with reward $R_{0.1}$, where $\Delta(\alpha, \beta) = 0.1$ in Eq. (7.2). The magenta graph represents the learning outcomes with reward $R_{0.125}$, where $\Delta(\alpha, \beta) = 0.125$, and the yellow graph represents the learning outcomes with reward $R_{0.145}$, where $\Delta(\alpha, \beta) = 0.145$. The black graph is the results got by swarm S_2 when learning was not involved.

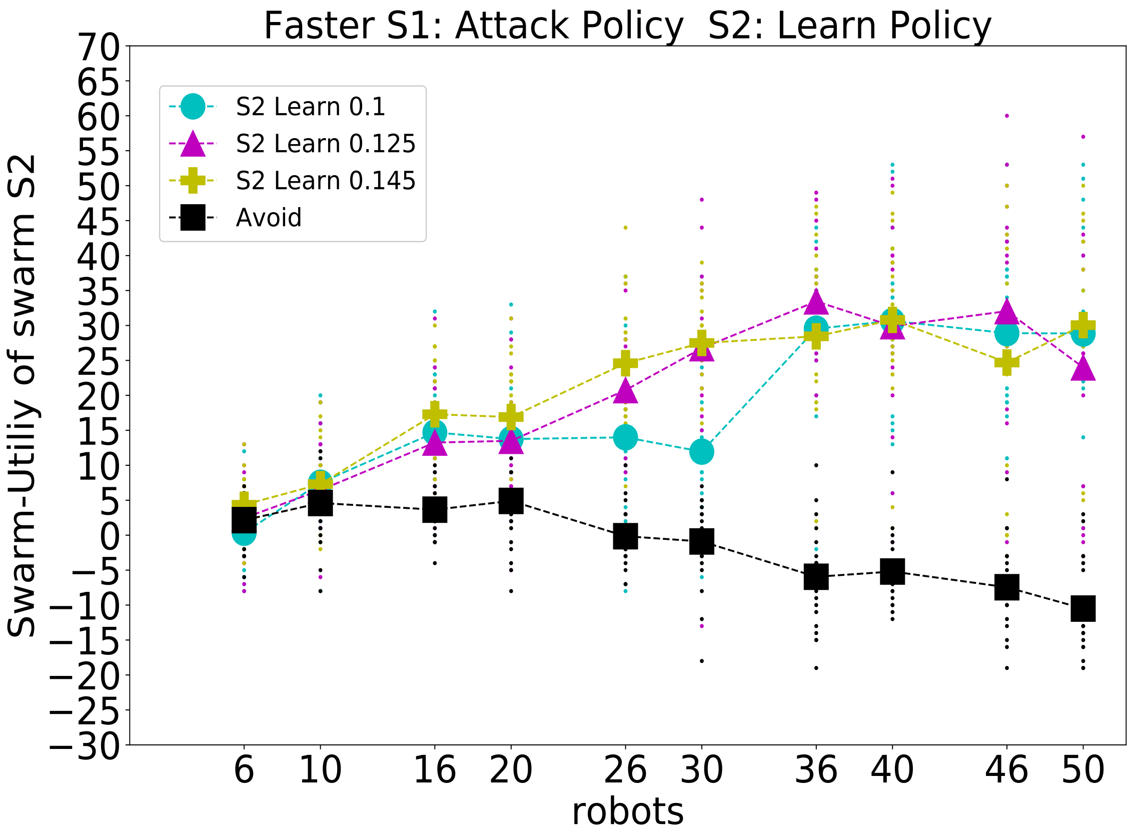


Figure 7.4: The SUD $\mathbb{U}_{S_2}$, of learning-swarm S_2, the plays versus faster, attacking swarm S_1.

In Figure 7.4, the learning swarm S_2, is the slower swarm, which moves at a speed of 60. The faster swarm S_1, which moves at a speed of 70, does not learn. The robots in this swarm always avoid colliding inside the swarm and attack robots that are not from their swarm. It can be seen that all the rewards tested overcome the non-learning swarm, as the SUD is greater than or equal to zero. In addition, the learning swarms get significantly better results than the non-learning swarm. At low densities all types of rewards achieved close results, while the learning with reward $R_{0.145}$ achieving the best results. It should be noted that reward $R_{0.1}$ at the lowest density (6 robots from both swarms in the field), achieved a swarm-reward of 0, i.e. did not defeat the previous swarm on average, but achieved the same result on average. At medium densities, (26 and 30 robots in the field), it can be seen that there is a very large gap between the types of learning. Learning with reward $R_{0.1}$ achieves significantly lower results from the other rewards, although it still defeats the second swarm. At high densities the various rewards again achieve relatively close results, where at any density it can be seen that there is different reward that achieves a higher result.

The results shown in the figure can be explained according to the Table 7.1. Each column refers to the percentage of player learned the attack policy, where its complement to 100% learned the avoid policy. The results are obtained using different $\Delta(\alpha, \beta)$ values of the reward function in Eq. (7.2), and their SUD comparison is shown in Fig. (7.4).

Number of players in game	Percentage of players learned to Attack with reward $R_{0.1}$	Percentage of players learned to Attack with reward $R_{0.125}$	Percentage of players learned to Attack with reward $R_{0.145}$
6	0%	0%	3%
10	1%	2%	3%
16	3.125%	5%	15%
20	3.5%	10%	25.5%
26	5.3%	30%	43.8%
30	12.33%	45.66%	53.66%
36	38.88%	60.55%	61.66%
40	53%	60.75%	62%
46	67.82%	60%	66%
50	66.4%	71.8%	71.2%

Table 7.1: Percentage of players from swarm S_2 learned to use *Attack* policy, in the experiment of Fig. (7.4).

When comparing the distribution of actions chosen by the robots at the end of the learning process, one can identify reasons for some of the differences or similarities in the SUD results obtained from the various rewards. At densities of 6 and 10 robots on the field, when the distribution of choices after the learning process is very close and low, it can be seen that the results are similar. Between densities of 16 to 26 robots on the field, higher results can be seen for a swarm with reward of $R_{0.145}$. It can be seen in the table that at these densities, the learning with this reward causes more robots to choose to attack opponents, and at these densities it allows better results. The largest difference in results between the tested rewards is for a density of 30 robots in the field. At this density, similar percentages (53% and 45%) of robots that learn to attack using rewards of $R_{0.125}$ and $R_{0.145}$, compared to only 12% of robots in the swarm that learn with reward $R_{0.1}$. The swarms that studied with a reward $R_{0.125}$ and $R_{0.145}$ received an average swarm-reward of 27, compared to the swarm that studied with a reward of $R_{0.1}$ that received an average of only 13. Therefore, it can be said that at this density when half of the swarm is attacking, the result is twice as good than the case where only a minority learns to attack. In the high densities it can be seen that the percentages are closer and so are the results.

7.2.3 Learning Swarm vs. a Faster Avoiding-Swarm

We now apply learning in order to successfully overcome the learning-swarm's disadvantage of speed as described earlier in this section. The policy of the non-learning faster swarm is Avoid, i.e, flight from all robots of both swarms. Fig. 7.3 provides the result of one of the examples mentioned in Subsection 7.1, using reward of $R_{0.145}$.

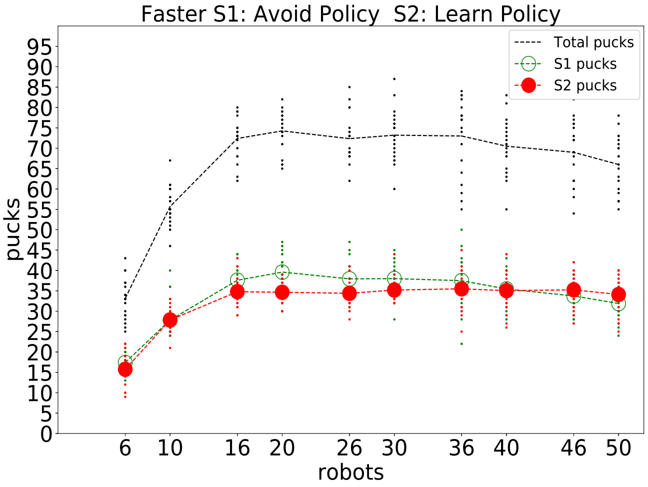


Figure 7.5: Swarm S_1 moves in 70 speed always avoids collisions, swarm S_2 move in 60 speed and learns how to act in collisions, using reward of $R_{0.145}$.

Figure 7.6 shows the results of the experiments run with the four types of rewards suggested above. The learning swarm is the slower swarm, which moves at a speed of 60. The fast swarm, which moves at a speed of 70, does not learn. The robots in this swarm always avoid collisions with all types of robots. The SUD of the experiment shown in Fig. 7.3 is represented by the yellow line in Fig. 7.4.

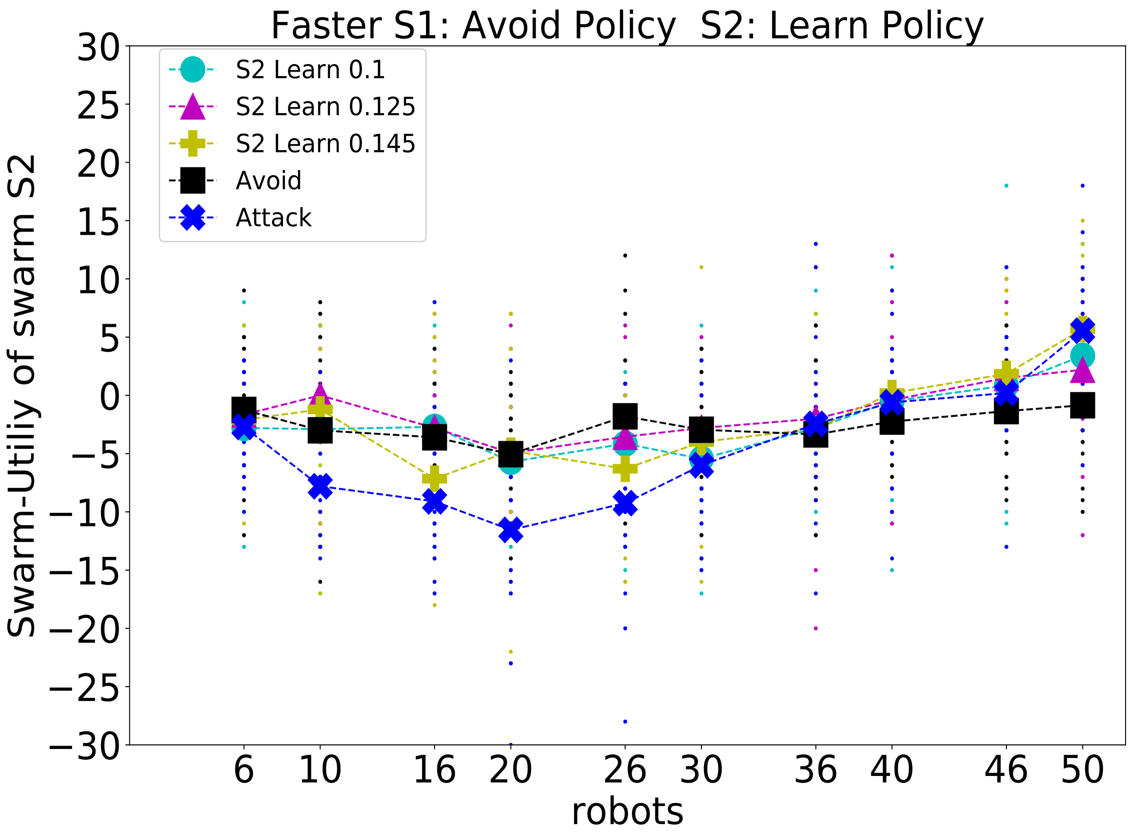
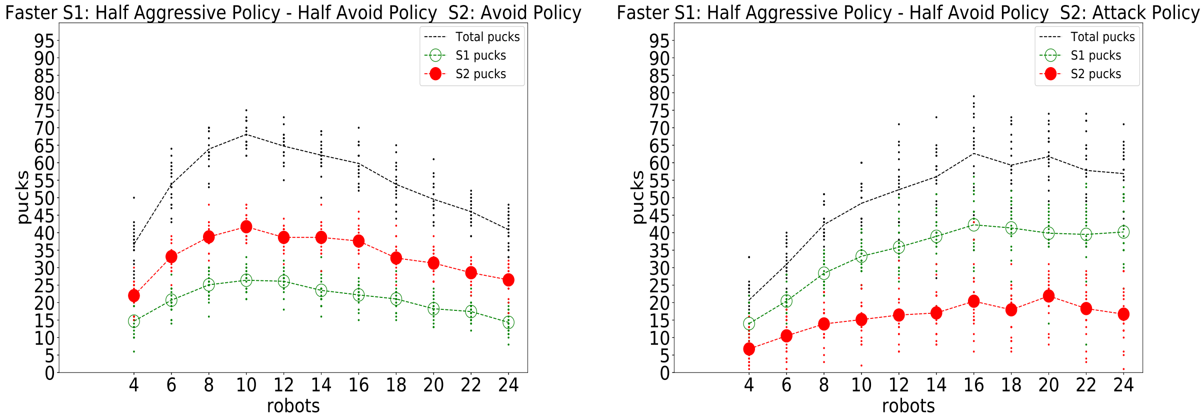


Figure 7.6: The SUD $\mathbb{U}_{S_2}$, of learning-swarm S_2, that competes a faster swarm S_1, that avoids collisions.

The blue and the black lines represents the non-learning behavior which presented here as a reference. Although the differences of the swarm-utilities got by the different rewards were very minor, one can observe that the learners' swarm-utilities in all learning cases, follow the black line for lower densities, where it holds the advantage, and follow the blue line in higher densities, where it holds the advantage. There is no learning that achieves significantly better results. In this case, there are some densities where the learning does not seem to achieve better results than in a fixed policy, even though there are some densities where the learning policy gets higher results. Our assumption is that the competing swarm is a very strong swarm; it has a speed advantage over the learning swarm, and it always avoids collisions and concentrates on collecting pucks. As long as it has the speed advantage, we could not find a competing swarm that could beat it. From these conclusions we thought test learning against another swarm when there is a speed advantage, which is easier to compete with, than the one described in this subsection, the mixed-swarm of the next subsection.

7.2.4 Learning Swarm vs. a Faster Mixed-Swarm

The avoiding swarm is really hard to defeat when having speed advantage in low densities. We wanted to compete against another faster, non-learning swarm, that would be at competent level between the attacking and avoiding swarms, so that we could test our learning model. We created a faster mixed-swarm, where half of the robots use *fight* method in any interaction; *Aggressive* policy, and the other half uses *flight* method in any interaction; *Avoid* policy.



(a) Swarm S_2 moves in 60 speed and always avoids collisions.

(b) Swarm S_2 moves in 60 speed and always attacks swarm S_1.

Figure 7.7: Number of pucks collected by two competitive swarms, in a 10 minutes limited game. S_1 is faster, half of its players attacks all players, and the other half avoids all players.

Fig. 7.7 shows the baseline, where no swarm is learning. In both figures 7.7a and 7.7b, swarm S_1 which is marked green, is faster and is mixed; half of the swarm uses *flight* method for all collisions, and the other half uses *fight* method for all collisions, as described above. Fig. 7.7a shows the case where the slower swarm S_2, marked red, uses *flight* method for all collisions, and Fig. 7.7b shows the case where swarm S_2 uses *flight* method for intra-swarm collisions and *fight* method for inter-swarm collisions. It is evident from the figure that a non-learning *avoid* policy is always better than the faster mixed-swarm, and the *attack* policy is always worse. Note that the number of pucks collected by the faster mixed-swarm, marked in green, which does not change policy between those two experiments, still has significant difference in performance, due to the behavior of the other swarm.

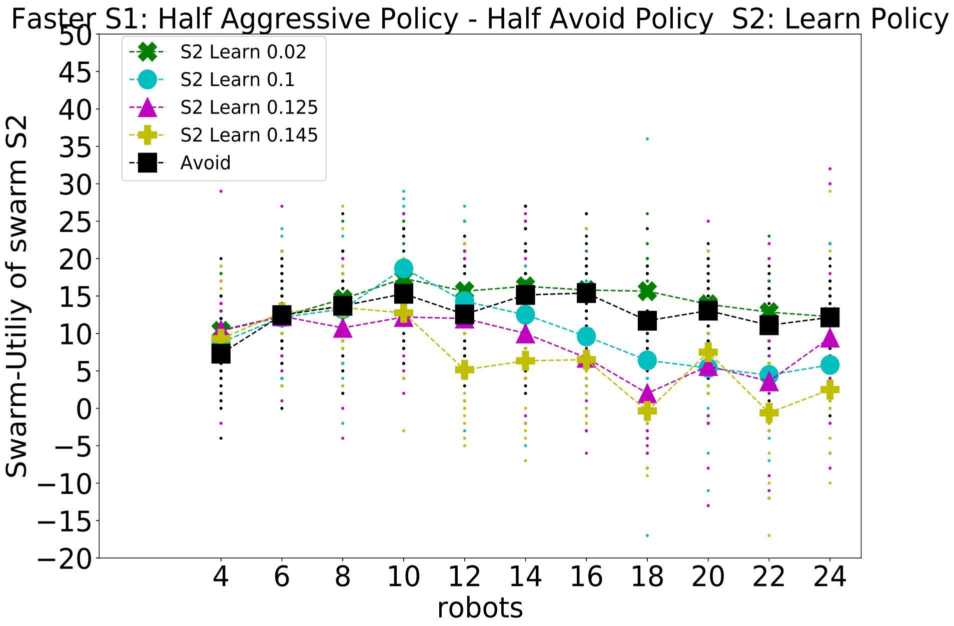


Figure 7.8: The SUD $\mathbb{U}_{S_2}$, of learning-swarm S_2, using different rewards, that competing with a faster mixed-swarm S_1.

Fig. 7.8 shows the swarm-utilities of the learning swarm S_2, when competing against a faster mixed-swarm, S_1, using different rewards parameter, $\Delta(\alpha)$. The black line represents the baseline of non-learning avoiding swarm shown in Fig. 7.7a. The colored lines represents learning with different rewards. One can observe that the learning did not achieve any significant advantage over the non-learning case. The only consistent small advantage was with the reward of $R_{0.02}$, that represents slow learning and have not shown any advantage in previous experiments. Therefore we can assume that the learning parameters may not fit this scenario. We hypothesize that learning during a competition with a mixed-swarm may benefit from treating the mixed-swarm as two different swarms. This hypothesis is examined in the next section, and shows significantly improved results. Unlike in this subsection, where the best learning could compare to a non-learning fixed policy baseline, learning while distinguishing between the different opponent swarms provided a higher score than any other case.

7.3 Three Swarms Competition: Learning with Speed Disadvantage

In the previous chapter we found that learning against a faster mixed-swarm, showed no better results than using a fixed policy. We hypothesize that learning during a competition with a mixed-swarm may benefit from treating the mixed-swarm as two different swarms. Therefore, we divided the faster mixed-swarm into two faster homogeneous different swarms with the same behavior described in the previous section. The major difference to all previous experiments, is the fact that when competing two swarms, each robot needs to learn a policy suitable for each competing swarm. Therefore, at the end of the learning process, each learning robot, will have a learned policy for each swarm-ID. To make the results comparable to previous experiments, the number of robots in the learning swarm is equal to the total number of robots in the two other faster non-learning swarms.

Figure 7.9 shows an example of an arena with robots from the three swarms. Each swarm was given its own ID and color so that all robots could be distinguished. The learning swarm is marked in red LEDs, the attacking swarm in yellow, and the avoiding swarm in green. The total amount of non learning robots, i.e. yellow and green robots is equal to the number of learning robot, i.e. red robots, as explained above.

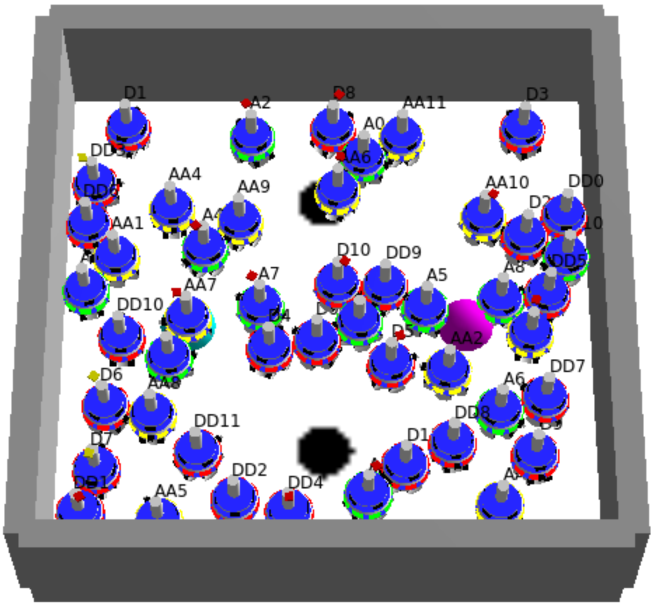


Figure 7.9: Three swarms on the field, swarm-ID is visible with red, green and yellow LEDs.

We examine two different cases. We first let the red swarm learn with the same reward function and parameters for both swarms, and the learning process is separated for each swarm. It means that at the end of the learning period, each learning robot can learn a different policy for each competing swarm. The results of this case are shown in Fig. 7.10. We compare the SUD of the learning swarm S_2, which is defined in Eq. 3.7. Since we measure in the experiments the pucks collected by each swarm, the SUD measured in this figure is the total pucks collected by swarm S_2, minus the total pucks collected by swarm S_1 and S_3. We got way better results than the results of the learning process shown in Fig. 7.8. Furthermore, unlike the previous subsection, we also achieved significantly better results the fixed-policy non-learning case marked in black. The results got after a learning process are always equal or better than the results of a fixed-policy of Avoid, marked in black. The reason is obvious, each competing swarm affects the learning swarm differently, and therefore the learning process ends with different policies when it is split.

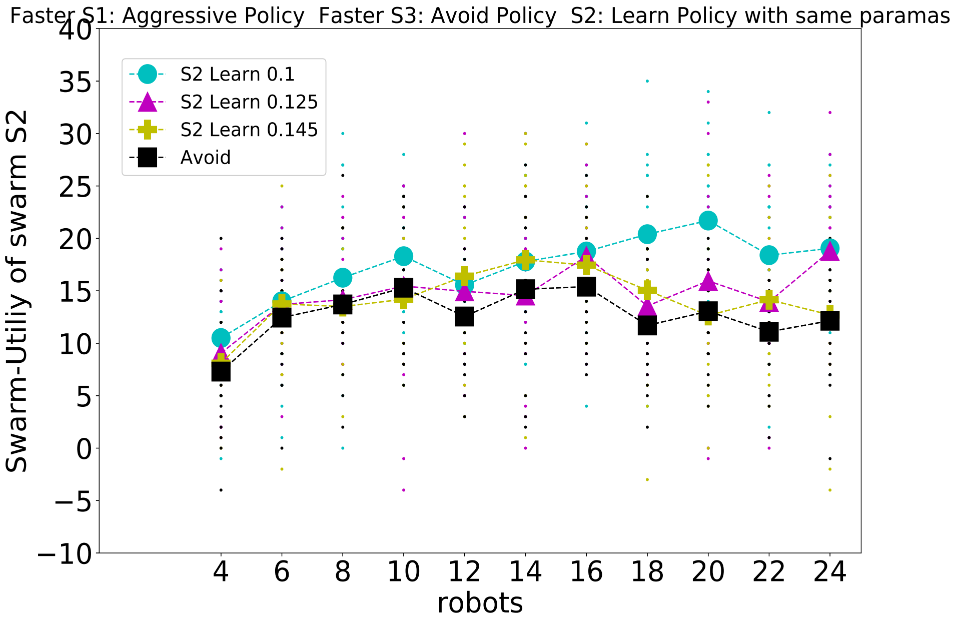


Figure 7.10: The SUD $\mathbb{U}_{S_2}$, of learning-swarm S_2, that competes faster swarms S_1 and S_3, using equal reward parameters.

We further let our swarm learn with different reward for each swarm. Since the α parameter quantifies the collision time with an opponent, we chose to use different α values during interactions with the different swarms. Unlike Fig. 7.8 where we used the same value of parameter α for both opponent swarms, we gave a fixed value for learning with swarm S_3, which uses *fight* policy in all interactions, and gave different values for reward of when interacting with swarm S_1, which uses *flight* method in all interactions.

The results of this case are shown in Fig. 7.11 and are better than any other run.

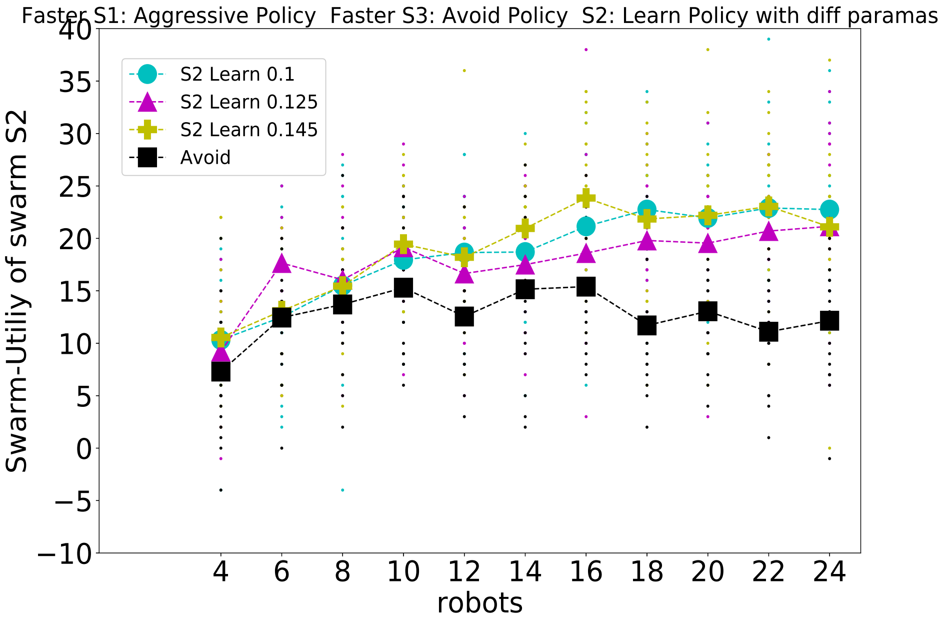


Figure 7.11: The SUD $\mathbb{U}_{S_2}$, of learning-swarm S_2, that competes faster swarms S_1 and S_3, using different reward parameters.

The conclusions from this section are follows. First, distinguishing between swarms with different behaviors, rather than treat them all as the same mixed-swarm is beneficial and brings significantly better results. During the learning phase each robot should maintain different learning processes, one for each swarm. This results with a separate policy for each swarm. Second, it is even better to use separate reward parameters for each swarm, to account for the different value gained by a robot by interacting with different types of opponents.

Now the question arises, are there optimal reward parameter values suitable for each type of swarm, and can it be learned by the individual learning robots during a competition. But this is left as future work.

7.4 Learning to Compete with a Larger Swarm

Another case of advantage to the non-learning swarm could be in the swarm size, since more players of the swarm could collect more pucks, or occupy the opponent swarm, and then clear the way for the rest of their swarm. To allow fair comparison with previous cases of swarm's advantage or disadvantage, we maintained the ratio of 6:7, used earlier. Figure 7.12 provides the baseline case where both swarm use the fixed policy of *Avoid*. We see a similar advantage in the resulted number of collected pucks as in previous cases. In this case, the total number of robots in the field (the horizontal axis), differs from other experiments, since the swarm sizes are unequal. So for 13 robots there were 7 robots for S_1 and 6 robots for S_2, and for 26, 39 and 52, there were double, triple and quadruple these numbers.

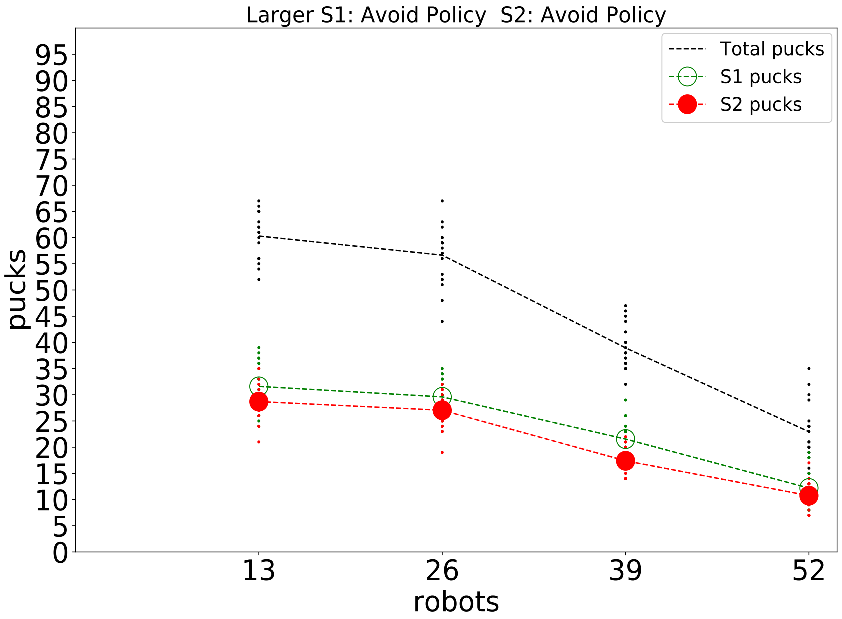
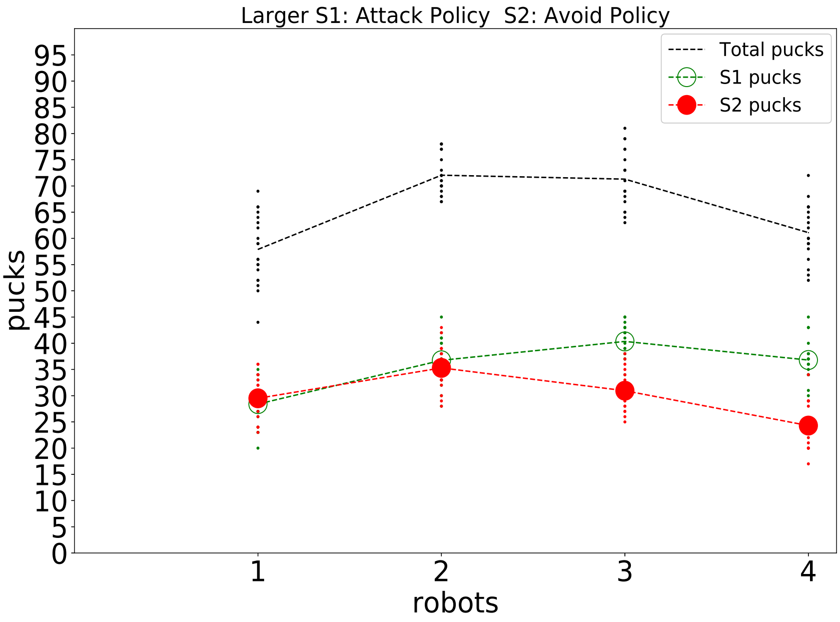
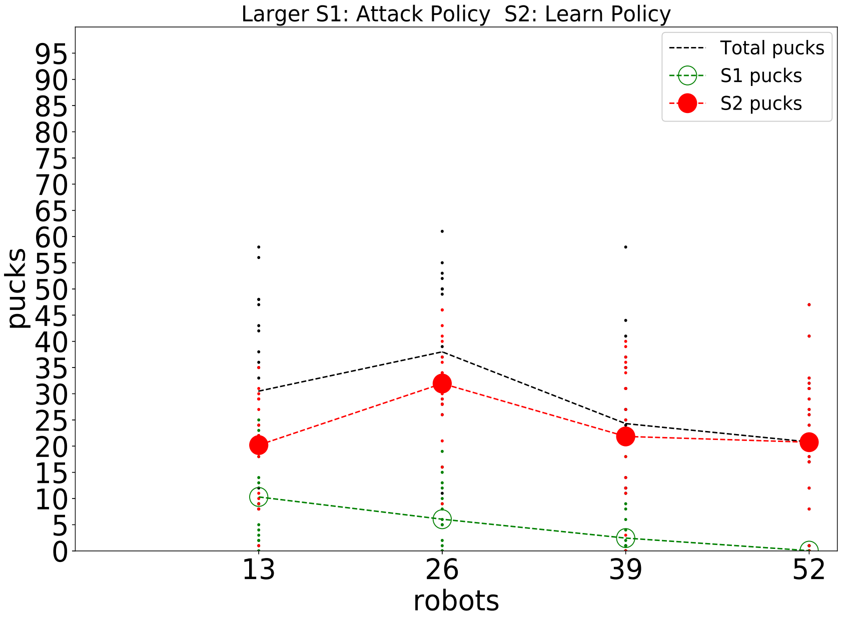


Figure 7.12: Swarm S_1, (marked in green) is larger than swarm S_2 (red), in ratio of 7:6.

Fig. 7.13 shows the case where the larger swarm S_1 uses the *Attack* policy. In Fig. 7.13a, the smaller swarm S_2 uses *Avoid* policy, and in Figure 7.13b, it learns the best policy. As expected, the non-learning larger swarm get equal or better scores when competing the smaller avoiding swarm. However, when learning is applied, the learning swarm defeats the attackers with significant gap, for all densities.



(a) Pucks collected when larger swarm S_1 Attacks, smaller swarmS_2 swarm Avoids.



(b) Pucks collected when larger swarm S_1 Attacks, smaller swarm S_2 learns best policies.

Figure 7.13: Number of pucks collected by two competitive swarms in a 10 minutes limited game with different densities, where swarm S_1 is larger than swarm S_2 in ration of 7:6.

To conclude, we showed another case where learning enables compensation of the disadvantaged swarm.

7.5 Learning to Compete in a Disadvantaged Arena

Another type of advantage to the non-learning swarm may be a shorter distance between the puck location and the homebase. Since the pucks distribution location and the homebase location is are unknown to the robots, but once a robot found a puck due to its random wandering in the arena, it finds the homebase by searching for its color. Therefore, decreasing the distance between the two, increases the probability to find the homebase earlier and therefore increase the expected value of the number of pucks in a given time. In the previous cases where the distance was equal, the beta of the two swarms, which is the contributor of the program time the robot's utility, were equal. In our case, since the distance is shorter, the same amount of time-unit of foraging, would contribute more on the expected value for bringing a puck to the homebase. Therefore we assume the beta values are different, with $\beta_1 > \beta_2$, where the larger β belongs to the swarm with the advantage.

In Figure 7.14, the top puck location is 16 cm closer to the green swarm's homebase, marked in cyan, than to the red swarm's homebase, marked in magenta. In order to make the results in this subsection comparable with the previous results in this chapter, the distance ratio of the locations of the upper puck from the two homebases, is 7:6. Therefore, the ratio used for reward calculation during the learning process was $\beta_1 : \beta_2 = 7 : 6$.

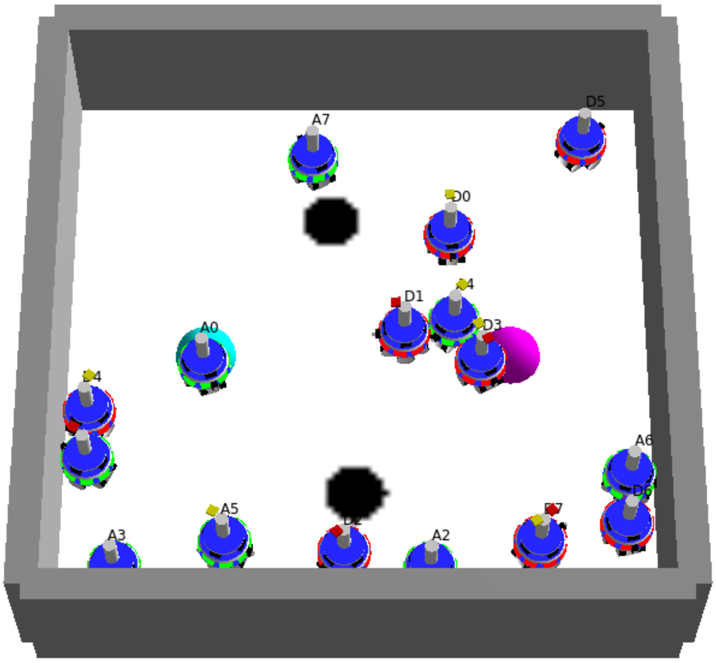


Figure 7.14: Upper puck is 16cm closer to S_1's homebase, marked in cyan, than to the S_2's homebase, marked in magenta.

The next two subsections examine the cases of competing with such advantaged swarm as described above. The first case is where the non-learning advantaged swarm uses *avoid* policy, and the second where the non-learning advantaged swarm uses *attack* policy.

7.5.1 Learning Swarm Competing with an Arena-Advantaged Avoiding-Swarm

In Figure 7.15 both swarms avoid collisions, and swarm S_1 has advantaged homebase location. It can be seen that the green swarm, S_1 has a consistent advantage over the red swarm, S_2, due to its homebase location advantage. We use this experiment as a baseline to compare with the influence of learning in such asymmetric environment.

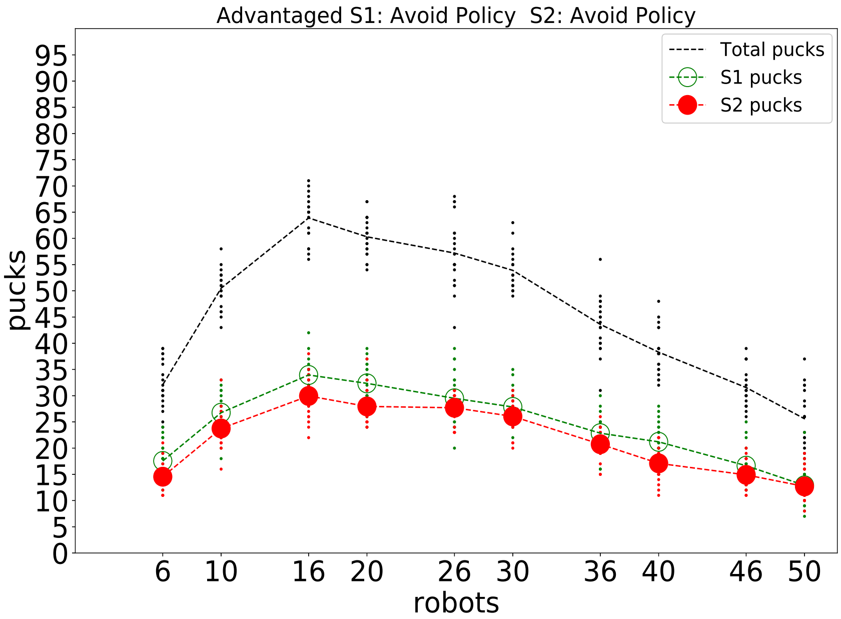
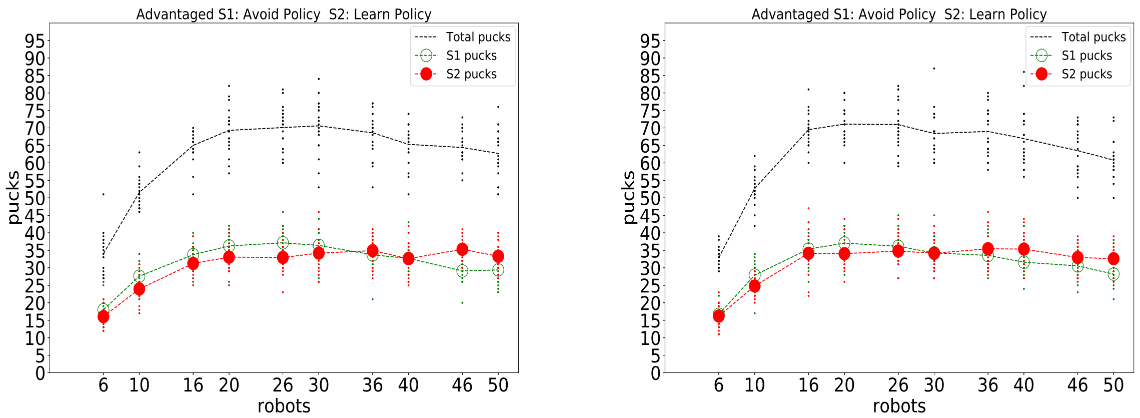


Figure 7.15: Swarm S_1 has an advantaged homebase location. Both swarms avoid collisions.

The following Figure 7.16 shows the results of learning with homebase location advantage, with two different rewards. The first learning process, shown in Figure 7.16a uses a reward an equal value of β parameters to both swarms. In other words, this reward does not take into account that for the same playing time, two players from two different swarms are likely to get a different score. The second figure, 7.16a, shows learning using a reward where the β parameter values are different, due to the advantage in the homebase location.



(a) Swarm S_1 has an advantaged homebase location. S_1 avoid collisions, S_2 learn how to act during collisions with reward based on $\beta_1 = \beta_2$.

(b) Swarm S_1 has an advantaged homebase location. S_1 avoid collisions, S_2 learn how to act during collisions with reward based on $\beta_1 : \beta_2 = 7 : 6$.

Figure 7.16: Number of pucks collected by two competitive swarms, in a 10 minutes limited game. S_1 has an advantaged homebase location.

The results shown in the figures above show that learning with equal beta values during the reward calculation, improves the non-learning case for higher densities. In this case the score is tied for densities of 36 and 40 robots, and it wins for 46 and 50 robots. After applying learning with two different betas in the reward calculation of the learning swarm, we see that it is tied on density of 30 robots, and wins for densities of 36 and above. In other words, using two different betas in the case of disadvantaged homebase location, improves the results of the learning swarm. Table 7.2 may explain the improvement when using different betas in the reward. Considering that the other swarm benefits more from program execution time unit, the learning swarm has higher incentive to interfere with the other swarm's robots rather than search for pucks. This is evident by the higher ratio of attack policy learned by robots in the learning swarm at the end of the learning process, when the reward is based on different beta values.

Number of players in game	Percentage of players learned to Attack with reward based on $\beta_1 = \beta_2$	Percentage of players learned to Attack with reward based on $\beta_1 : \beta_2 = 7 : 6$
6	1.66%	13.3%
10	1%	21%
16	1.8%	30%
20	3.5%	39.5%
26	13.4%	50%
30	25.6%	58.66%
36	53.8%	65.5%
40	53%	64.5%
46	61%	72%
50	67%	78.6%

Table 7.2: Percentage of players from swarm S_2 learned to use *Attack* policy, in the experiment of Fig. 7.4.

7.5.2 Learning Swarm vs. an Arena-Advantaged Attacking-Swarm

In this subsection, the non-learning advantaged swarm S_1 uses *attack* policy, i.e., uses *fight* method in inter-swarm interactions, and *flight* method in intra-swarm interactions. As a baseline shown in Figure 7.17, swarm S_2 uses *avoid* policy in all interactions, i.e, both swarms use fixed policies and do not learn. The results in the figure show that using these policies, swarm S_2 (red) collects more pucks in low densities even though it is in disadvantage, collects less pucks in higher densities.

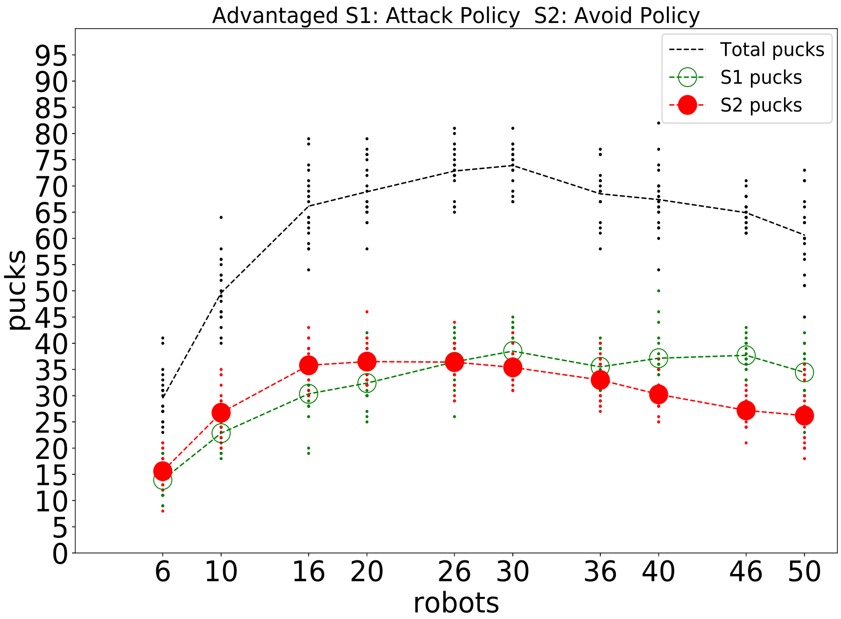
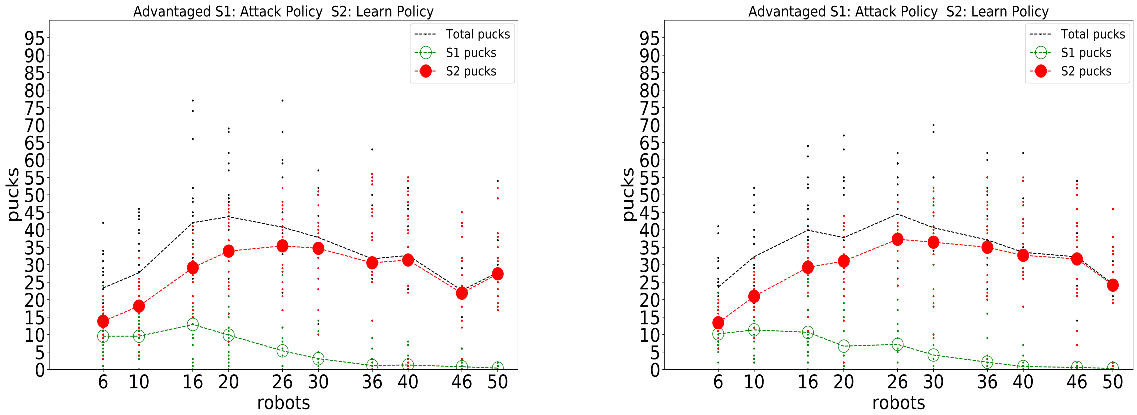


Figure 7.17: Swarm S_1 has an advantaged homebase location. Swarm S_1 attack enemies, swarm S_2 avoid collisions.

Similar to the previous subsection, the following Figure 7.18 shows the results of learning with homebase location advantage, with two different rewards. The first learning process, shown in Figure 7.18a uses a reward an equal value of β parameters to both swarms. In other words, this reward does not take into account that for the same playing time, two players from two different swarms are likely to get a different score. The second figure, 7.18b, shows learning using a reward where the β parameter values are different, due to the advantage in the homebase location.



(a) Swarm S_1 has an advantaged homebase location. S_1 attack enemies, S_2 learn how to act during collisions without considering the difference in β values.

(b) Swarm S_1 has an advantaged homebase location. S_1 attack enemies, S_2 learn how to act during collisions considering the difference in β values.

Figure 7.18: Number of pucks collected by two competitive swarms, in a 10 minutes limited game. S_1 is faster, half of its players attacks all players, and the other half avoids all players.

The results got by the swarm which applied learning based on $\beta_1 : \beta_2 = 7 : 6$ performed better when compared to the swarm which applied learning based on $\beta_1 = \beta_2$. The SUD (difference of pucks collected by the swarms) of the learning swarms with both equal and different beta values, is shown in Figure 7.19, and compared to the non-learning case of the fixed avoid policy. The cyan, magenta and black lines represent the swarm-utilities of the experiments shown in Fig. 7.18b, 7.18a and 7.17 respectively. As in the previous subsection, we assume that the improvement is a result of a more incentive to the learning swarm to interfere with the other swarm's robots rather than search for pucks. It can be seen that although the change in the β parameter is 6:7 which changes the program time influence in 11:12, there is still some consistent advantage of changing the β.

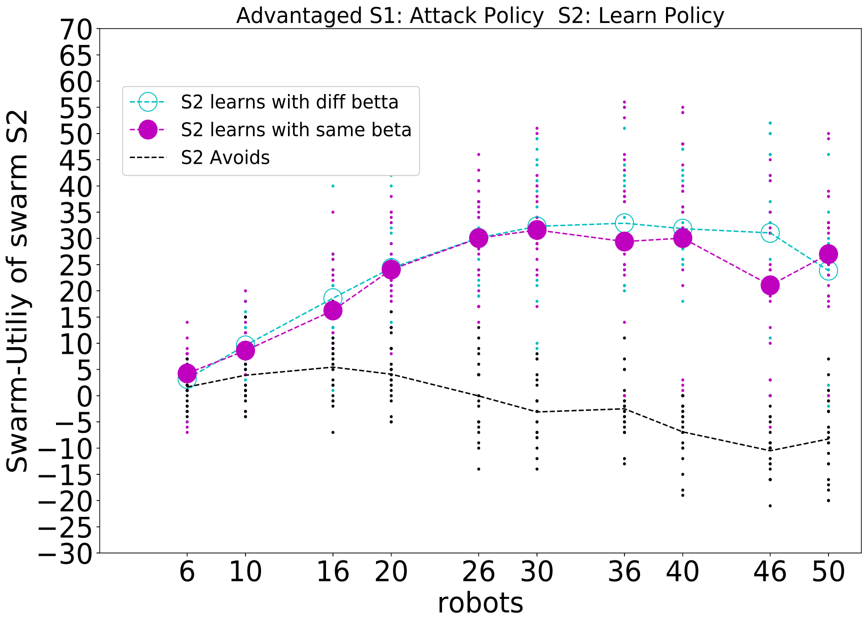


Figure 7.19: The SUD of swarm S_2 in both learning processes, where Swarm S_1 has an advantaged homebase location and uses attack policy. The magenta SUD represents the equal beta case, and the cyan represents the different beta case.

To conclude, in this chapter we examined different cases where the learning swarm had to overcome a disadvantage when compared to its competitor swarm. We showed cases of disadvantage in speed, number of robots per swarm, and in homebase location in the arena. In all cases, we found a potential way to quantify the advantage of the other swarm into the reward function during learning, and showed a clear improvement as a result of the learning with these reward functions. We additionally showed the case of three swarm competition where the two other faster non-learning swarms different policies. In this case, we showed that the learning swarm should maintain two different learning processes, one for each swarm, where every process has its own alpha and beta parameters, based on the policies of the specific opponent swarm.

Chapter 8

Discussion

In this chapter, we discuss some of the gaps between the theory and its application, some potential ideas that have not been fully proofed in this work, and future proposed work.

Our model suggests that the global density of each swarm influences the individual and swarm utilities, and therefore should be taken into consideration in the reward calculation. However, since the robots' sensing and communication capabilities are local, and there is no global knowledge, the robots cannot be directly aware of the total amount of players, or the actual density in the arena. Therefore, in the reward approximation in Chapter 5, we use the local density rather than the global density as an estimator. The local density is limited by the number of sensors on a robot, and thus, is affected only by its immediate surroundings. This change is supported by the experiments' results, but is not theoretically explained.

The theoretical model, and its estimations and approximations, presented in Chapters 3 and 4 were related to a general case of multi-swarm competitions. Therefore, both the individual and the swarm utilities were dependent on the program execution and interaction times, $T(p), T(a)$. When applying the model to the more specific task of foraging, the intuitive utility measure would be the number of pucks collected rather than some functions of times, as presented in the figures. However, the individual robot has used the time related utility as the basis of its learning process, which still resulted in increasing the number of collected pucks for the learning swarm. The relation between time related swarm utility and number of pucks collected has not been fully theoretically proven. Even if such relation exists, it would fit a specific system design, and not the general problem.

Another potential gap in the simulation used in this work is the definition of spatial interaction by an individual robot. In this work only if the front sensor identifies another player in a predetermined distance, the robot enters an interaction state. Therefore, there might be cases where robots are not aware when being interrupted, e.g., when hit from the side. Our reward approximation tries to compensate for this gap by counting the number of immediate surrounding robots during interactions. This might be a possible explanation to the fact that in Figure 6.3 the attacking swarm counted its time of inter-swarm interaction almost doubled the amount of time counted by the avoiding swarm.

When trying to estimate the individual's effect on its surrounding, we assumed in Subsection 4.1.2 that in the absence of player i, if there was no conflict between the players, player j would be in Program-Time instead of in Action Time. Therefore, it should get the utility accordingly. This assumption has not been proved or tested. A possible way to explore this assumption is to run simulations without a single player and compare the differences in the action and program times of the other players, and on the scores got by each swarm.

An unsolved question arises from some of the experiments, namely, is there an optimal value of the model parameters α and β, depending on the density and the type of the opponent. A clue could be found in Figure 6.12 where better results happened for different densities for different values of α. Another example is the difference in the results between Figures 7.10 and 7.11 in a three swarm competition. In the first experiment, the learning parameters were two learning process occurred with identical parameters for each opponent swarm. The second experiment suggested different α parameters for each swarm, and got better results. Therefore, the it might be assumed that the model parameters may be dependent on both the density and the type of the opponent. Moreover, even though we have demonstrated advantage of specific parameter values for a given scenario, we did not look for the optimal value will provide the best results. We leave such exploration for future work, and propose to let the individual robots learn those values during the learning process.

Although the game-theoretical model is generic, our experiments concentrated on learning to resolve spatial interaction in a way that would maximize the Swarm Utility Difference (SUD). In chapter 7, when exploring cases of learning with disadvantage, we have encountered some cases where there is no action policy in interaction time, that would be able to compensate for the disadvantage during program time. Specifically, when the advantaged non-learning swarm uses Avoid policy, there is no possibility for the learning swarm to interfere strongly enough to win the game.

We propose for future work to examine the applicability of the theoretical model to tasks other than foraging which was used in this work, to validate the generalization of the proposed model.

Chapter 9

Conclusion

In this work we developed a general game-theoretic model for multiple competitive swarms. We formulated the individual and global utilities for K-swarms competition, with a single assumption of zero-sum game between two individual players from different swarms. We defined the term Swarm Utility Difference (SUD), which is a measure of scoring the game results for multiple swarms. We showed that the game between two swarms is always a zero-sum game, and showed possible extension to zero-sum game for the K-swarm case. We calculated the influence of individual players on all swarms-utilities, by considering the effect of their presence and absence.

We then applied the theory into the field of competitive robot swarms. We defined the global and individual utilities as a function of time for a generic multi-swarm competition, by estimating the individual player's utility and the effect on its surroundings, based on WLU function. One nontrivial observation from the model, is that a robot can increase its SUD not only by performing its original task, but also by interfere its opponents in performing their tasks.

We also proposed a learning process of each individual robot in multi-swarm competition, by calculating its own reward, and providing a general way for evaluation and selection of its possible actions. To find such interaction policies, we apply reinforcement learning methods to train robots individually. The robots cannot communicate with each other. Thus, global information such as the collective score of a swarm or the total number of members in each swarm, is unknown to any of its member agents. The proposed learning model tries to overcome the gap due to this partial information known to each robot, by considering the swarm identity of the other robots during each interaction, and approximating differences between the swarms.

As an example, we then applied the general model for the more specific case of multi-swarm competitive foraging. We also defined relevant action sets and policies, and proposed an experimental environment and settings. We examined the model on the challenging problem of how robots in a competitive multi-swarm environment should interact during spatial conflicts, in order to outperform the other swarms.

We have ran an extensive series of simulated experiments, and validated the proposed model. The experiments mainly concentrate on competitive multi-swarm cases, but also generalized for the single swarm case, to compare with previous works and show applicability. We checked the effect of learning on symmetric two swarm competition, where all players perform identically during their main task execution, but may act differently during interacting with other robots. We showed that the learning swarms performed equally or better than a non-learning swarms, which uses a predefined policy. We explored the performance of the learning model in different environments and hyper parameters.

We then expanded the simulation for cases of a learning swarm with initial disadvantage. We explored the cases of competing a faster swarm, an uneven number of robots in the swarms, and a case of advantaged location of homebase in the arena. When in disadvantage, being equal to the opponent is insufficient to win, and the learning swarm needs to overcome the opponent's advantage gap. To do so, we modified the reward function from the symmetric case to the asymmetric case, to reflect the overcome the disadvantages. We also explored the case of three swarms competition and showed generalization. All results support efficacy with the proposed model.

Bibliography

[1] Noa Agmon, Sarit Kraus, and Gal A Kaminka. Multi-robot perimeter patrol in adversarial settings. In *IEEE International Conference on Robotics and Automation*, pages 2339–2345. IEEE, 2008.

[2] Gürdal Arslan, Jason R Marden, and Jeff S Shamma. Autonomous vehicle-target assignment: A game-theoretical formulation. *ASME*, 2007.

[3] Ioannis Arvanitakis and Anthony Tzes. Collaborative mapping and navigation for a mobile robot swarm. In *25th Mediterranean Conference on Control and Automation (MED)*, pages 696–700. IEEE, 2017.

[4] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

[5] Levent Bayındır. A review of swarm robotics tasks. *Neurocomputing*, 172:292–321, 2016.

[6] Gerardo Beni. From swarm intelligence to swarm robotics. In *International Workshop on Swarm Robotics*, pages 1–9. Springer, 2004.

[7] Gerardo Beni. Swarm intelligence. *Complex Social and Behavioral Systems: Game Theory and Agent-Based Models*, pages 791–818, 2020.

[8] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[9] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[10] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, pages 183–221, 2010.

[11] Yang Cai, Ozan Candogan, Constantinos Daskalakis, and Christos Papadimitriou. Zero-sum polymatrix games: A generalization of minmax. *Mathematics of Operations Research*, 41(2):648–655, 2016.

[12] Mitch Campion, Prakash Ranganathan, and Saleh Faruque. A review and future directions of uav swarm communication architectures. In *2018 IEEE international conference on electro/information technology (EIT)*, pages 0903–0908. IEEE, 2018.

[13] Eduardo Castello, Tomoyuki Yamamoto, Fabio Dalla Libera, Wenguo Liu, Alan FT Winfield, Yutaka Nakamura, and Hiroshi Ishiguro. Adaptive foraging for simulated and real robotic swarms: the dynamical response threshold approach. *Swarm Intelligence*, 10(1):1–31, 2016.

[14] Vishnu S Chipade and Dimitra Panagou. Multi-swarm herding: Protecting against adversarial swarms. In *59th IEEE Conference on Decision and Control (CDC)*, pages 5374–5379. IEEE, 2020.

[15] Timothy H Chung, Kevin D Jones, Michael A Day, Marianna Jones, and Michael Clement. 50 vs. 50 by 2015: Swarm vs. swarm uav live-fly competition at the naval postgraduate school. 2013.

[16] Alireza Dirafzoon and Edgar Lobaton. Topological mapping of unknown environments using an unlocalized robotic swarm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5545–5551. IEEE, 2013.

[17] Khiem N Doan, An T Le, Than D Le, and Nauth Peter. Swarm robots' communication and cooperation in motion planning. In *Mechatronics and*

Robotics Engineering for Advanced and Intelligent Manufacturing, pages 191–205. Springer, 2017.

[18] Yinon Douchan, Ran Wolf, and Gal A Kaminka. Swarms can be rational. In *AAMAS*, pages 149–157, 2019.

[19] Yotam Elor and Alfred M Bruckstein. Autonomous multi-agent cycle based patrolling. In *International Conference on Swarm Intelligence*, pages 119–130. Springer, 2010.

[20] Kala Garapati, Juan Jesús Roldán, Mario Garzón, Jaime del Cerro, and Antonio Barrientos. A game of drones: Game theoretic approaches for multi-robot task allocation in security missions. In *Iberian robotics conference*, pages 855–866. Springer, 2017.

[21] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for non-stationary bandit problems. *arXiv preprint arXiv:0805.3415*, 2008.

[22] Eden R. Hartman. Swarming bandits: A rational and practical model of swarm robotic tasks. Master's thesis, Bar Ilan University, 2022.

[23] Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*, 2017.

[24] Tien-Ruey Hsiang, Esther M Arkin, Michael A Bender, Sándor P Fekete, and Joseph SB Mitchell. Algorithms for rapidly dispersing robot swarms in unknown environments. In *Algorithmic Foundations of Robotics V*, pages 77–93. Springer, 2004.

[25] Jiangping Hu and Wei Xing Zheng. Emergent collective behaviors on coopetition networks. *Physics Letters A*, 378(26-27):1787–1796, 2014.

[26] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[27] Gal A Kaminka, Dan Erusalimchik, and Sarit Kraus. Adaptive multi-robot coordination: A game-theoretic perspective. In *IEEE International Conference on Robotics and Automation*, pages 328–334. IEEE, 2010.

[28] Daniela Kengyel, Heiko Hamann, Payam Zahadat, Gerald Radspieler, Franz Wotawa, and Thomas Schmickl. Potential of heterogeneity in collective behaviors: A case study on heterogeneous swarms. In *International conference on principles and practice of multi-agent systems*, pages 201–217. Springer, 2015.

[29] James Kennedy. Swarm intelligence. In *Handbook of nature-inspired and innovative computing*, pages 187–219. Springer, 2006.

[30] Jong-Hyun Lee, Chang Wook Ahn, and Jinung An. A honey bee swarm-inspired cooperation algorithm for foraging swarm robots: An empirical analysis. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 489–493. IEEE, 2013.

[31] Seoung Kyou Lee, Sándor P Fekete, and James McLurkin. Structured triangulation in multi-robot systems: Coverage, patrolling, voronoi partitions, and geodesic centers. *The International Journal of Robotics Research*, 35(10):1234–1260, 2016.

[32] Tao Li, Guanze Peng, Quanyan Zhu, and Tamer Başar. The confluence of networks, games, and learning a game-theoretic framework for multiagent decision making over networks. *IEEE Control Systems Magazine*, 42(4):35–67, 2022.

[33] Michael L Littman. Friend-or-foe q-learning in general-sum games. In *ICML*, pages 322–328, 2001.

[34] Qi Lu, G Matthew Fricke, John C Ericksen, and Melanie E Moses. Swarm foraging review: Closing the gap between proof and practice. *Current Robotics Reports*, pages 1–11, 2020.

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

on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, pages 6466–6471. IEEE, 2009.

[36] Dov Monderer and Lloyd S Shapley. Potential games. *Games and economic behavior*, 14(1):124–143, 1996.

[37] Daniel Morgan, Soon-Jo Chung, and Fred Y Hadaegh. Model predictive control of swarms of spacecraft using sequential convex programming. *Journal of Guidance, Control, and Dynamics*, 37(6):1725–1740, 2014.

[38] Jianjun Ni, Guangyi Tang, Zhengpei Mo, Weidong Cao, and Simon X Yang. An improved potential game theory based method for multi-UAV cooperative search. *IEEE Access*, 8:47787–47796, 2020.

[39] Ann Nowé, Peter Vrancx, and Yann-Michaël De Hauwere. Game theory and multi-agent reinforcement learning. In *Reinforcement Learning*, pages 441–470. Springer, 2012.

[40] John Oyekan and Huosheng Hu. Ant robotic swarm for visualizing invisible hazardous substances. *Robotics*, 2(1):1–18, 2013.

[41] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, et al. Argos: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm intelligence*, 6(4):271–295, 2012.

[42] Juan Jesús Roldán, Jaime Del Cerro, and Antonio Barrientos. Should we compete or should we cooperate? applying game theory to task allocation in drone swarms. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5366–5371, 2018.

[43] Avi Rosenfeld, Gal A Kaminka, Sarit Kraus, and Onn Shehory. A study of mechanisms for improving robotic group performance. *Artificial Intelligence*, 172(6-7):633–655, 2008.

[44] Joseph A Rothermich, M İhsan Ecemiş, and Paolo Gaudiano. Distributed localization and mapping with a robotic swarm. In *International Workshop on Swarm Robotics*, pages 58–69. Springer, 2004.

[45] Samuel Rutishauser, Nikolaus Correll, and Alcherio Martinoli. Collaborative coverage using a swarm of networked miniature robots. *Robotics and Autonomous Systems*, 57(5):517–525, 2009.

[46] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *International workshop on swarm robotics*, pages 10–20. Springer, 2004.

[47] Erol Sahin and Alan FT Winfield. Special issue on swarm robotics. *Swarm Intell.*, 2(2-4):69–72, 2008.

[48] Moshe N Samson and Noa Agmon. Competitive coverage:(full) information as a game changer. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6633–6640. IEEE, 2020.

[49] Thomas Schmickl, Ronald Thenius, Christoph Moslinger, Jon Timmis, Andy Tyrrell, Mark Read, James Hilder, Jose Halloy, Alexandre Campo, and Cesare Stefanini. Cocoro–the self-aware underwater swarm. In *IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pages 120–126. IEEE, 2011.

[50] Hazim Shakhatreh, Abdallah Khreishah, Jacob Chakareski, Haythem Bany Salameh, and Issa Khalil. On the continuous coverage problem for a swarm of uavs. In *IEEE 37th Sarnoff Symposium*, pages 130–135. IEEE, 2016.

[51] Dylan A Shell and Maja J Mataric. On foraging strategies for large-scale multi-robot systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2717–2723. IEEE, 2006.

[52] Daigo Shishika, Katarina Sherman, and Derek A Paley. Competing swarms of autonomous vehicles: Intruders versus guardians. In *Dynamic Systems and Control Conference*, volume 58288, page V002T14A006. American Society of Mechanical Engineers, 2017.

[53] Adam Slowik and Halina Kwasnicka. Nature inspired methods and their industry applications-swarm intelligence algorithms. *IEEE Transactions on Industrial Informatics*, 14(3):1004–1015, 2017.

[54] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[55] Mohamed S Talamali, Thomas Bose, Matthew Haire, Xu Xu, James AR Marshall, and Andreagiovanni Reina. Sophisticated collective foraging with minimalist agents: a swarm robotics test. *Swarm Intelligence*, 14(1):25–56, 2020.

[56] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[57] Ying Tan and Zhong-yang Zheng. Research advance in swarm robotics. *Defence Technology*, 9(1):18–39, 2013.

[58] Michael Ummels. *Stochastic multiplayer games: Theory and algorithms*. Amsterdam University Press, 2010.

[59] Janardan Kumar Verma and Virender Ranga. Multi-robot coordination analysis, taxonomy, challenges and future scope. *Journal of Intelligent & Robotic Systems*, 102(1):1–36, 2021.

[60] Tamás Vicsek and Anna Zafeiris. Collective motion. *Physics reports*, 517(3-4):71–140, 2012.

[61] Gerhard Weiss. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999.

[62] Alan FT Winfield. Foraging robots. 2009.

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

[64] Erfu Yang and Dongbing Gu. Multiagent reinforcement learning for multi-robot systems: A survey. *In Technical report, University of Essex Technical Report CSM-404, Department of Computer Science*, 2004.

[65] Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*, 2020.

[66] Roi Yehoshua, Noa Agmon, and Gal A Kaminka. Robotic adversarial coverage of known environments. *The International Journal of Robotics Research*, 35(12):1419–1444, 2016.

[67] Ouarda Zedadra, Nicolas Jouandeau, Hamid Seridi, and Giancarlo Fortino. Multi-agent foraging: state-of-the-art and research challenges. *Complex Adaptive Systems Modeling*, 5(1):1–24, 2017.

[68] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pages 321–384, 2021.

[69] Haitao Zhao, Hai Liu, Yiu-Wing Leung, and Xiaowen Chu. Self-adaptive collective motion of swarm robots. *IEEE Transactions on Automation Science and Engineering*, 15(4):1533–1545, 2018.