Turk J Elec Engin, VOL.15, NO.2 2007, © TÜBİTAK

Swarm Robot Systems Based on the Evolution of Personality Traits

Sidney Nascimento GIVIGI Jr., Howard M. SCHWARTZ
Department of Systems and Computer Engineering, 1125 Colonel By Drive, Carleton University; Ottawa, ON, K1S 5B6, CANADA
e-mail: sgivigi@sce.carleton.ca

Abstract

Game theory may be very useful in modeling and analyzing swarms of robots. Using game theory in conjunction with traits of personalities, we achieve intelligent swarm robots. Traits of personality are characteristics of each robot that define the robots' behaviours. The environment is represented as a game and due to the evolution of the traits through a learning process, we show how the robots may react intelligently to changes in the environment. A proof of convergence for the proposed algorithm is offered. The process of selection of traits is discussed and the potential of the modeling is demonstrated in several different simulations.

1. Introduction

SWARM intelligence may be defined as a distributed problem-solving technique for multi agent dynamic systems based on self-organization theory and inspired by collective social behaviour [1]. Moreover, we could add that the main natural (or biological) base for swarm intelligence has been the social insect colonies, for they are clearly simple individually, but present a highly complex social behaviour based on (supposedly) very simple rules [2]. Also, they have almost no direct communication, but are able to overcome difficult tasks by observing changes in the environment performed by other individuals [1, 2].

Swarm robotics has been defined by Sahin as *the study of how large number of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions among agents and between the agents and the environment* [3]. In our approach we use a looser definition in what concerns the term *desired* collective behaviour. We are more interested in what Beni [4] called *unpredictability* of the system. In other words, we are interested in the patterns and behaviours that will arise from the group interactions.

An important work that may be cited on swarm robotics was performed by Dorigo et. al. [5]. In this paper, they provide an overview of their project (SWARM-BOTS). They report experimental work that show how robots may cooperate and coordinate tasks. The main difference with our approach is in the definition of the relationship among robots and the environment. In this article we make extensive use of Game Theory.

In practice, swarm intelligence is not intended to generate a rational individual entity, as classical Artificial Intelligence proposes [1]. However, through investigation of simple computational models that

may be implemented in simple machines, swarm intelligence tries to explore how complex tasks might be performed by a social entity due to behaviours that are not directly predicted by the particular characteristics of each individual [2]. For example, if we have a society with a majority of peaceful agents, when some aggressive individuals enter the community, we cannot say that the majority will make the new individuals become peaceful. It is our purpose to try to predict such behaviours through modeling and simulation in order to justify techniques that may be used in a swarm-based robotic environment.

For the purpose of this work, we defined the following design guidelines:

- The algorithms the robots must execute are simple (computationally) and may be run by simple DSP (or even FPGA) chips.
- The robots sensory capacity is limited and supposed to be the same.
- The message exchanging is very limited or nonexistent. When (and if) the robots communicate with each other, no guarantee is given that the message will be received or, if it is received, that the content of the message was understood.
- The robots are able to sense the presence of other robots and, for simulation sake, the position of targets or enemies.

In order to observe the emergence of group behaviours, we rely on the concept of "personality traits" [6, 7]. Through the adaptation of personality traits the underlying behaviour of each robot changes and by changing each robot's behaviour the group's behaviour also changes [1].

The environment is modelled as a game and techniques of learning in games are used. The field of learning has been extensively investigated. For example, [8] brings a comprehensive discussion of the theory behind learning in games. In [9, 10, 11], we find discussions related to the topic as well as very interesting simulations of possible applications. We use a version of fictitious play in this work. This learning technique was introduced in [12] and its convergence was proven in [13]. In addition to fictitious play (used in the case of a zero-sum game and derived from our general algorithm), we also make use of reinforcement learning, a very powerful tool with several different applications to the field of artificial intelligence [14].

This work is divided as follows. *Section 2* is a brief discussion of Game Theory from the point of view of social modeling. *Section 3* introduces the concept of personality traits and how they evolve. *Section 4* reports the simulations that clarify the ideas discussed. Finally, *section 5* concludes the paper with a discussion of all simulations, bringing them together in a detailed fashion.

2. Dynamic modeling using game theory

One of the major successes in the field of economics and social sciences in the past decades has been the application of Game Theory to the modeling of social interactions of rational entities for the prediction of outcomes of conflicts among them [15]. It turns out that the same approach may be used in the modeling of robot swarms, since their formation may be thought of as a social interaction of individuals [1].

Game Theory can be defined as "the study of mathematical models of conflict and cooperation between intelligent rational decision-makers" [15]. Therefore, it seems natural to explore this technique in order to represent the behaviour of robots, since robots may be regarded as "intelligent rational decision-makers". Although in our case their rationality is limited, this does not impair the application of the theory. For

example, evolutionary models have been developed using Game Theory where, obviously, the agents involved cannot be regarded as "intelligent rational" entities [16] and the situations they usually are involved in concern conflict and cooperation [17].

One very important thing to notice is what is meant by conflict. Conflict does not mean fight or engagement and does not presuppose an enemy. Even teammates have conflicts and even one single individual has conflicts. It is not our intention to analyze conflicts from a philosophical point of view, but we do not restrain ourselves on the usual definition of conflict as the fight between contraries. For our purposes, a conflict is established when one trait of personality leads to a different action than another trait of personality or when one robot has individual interests that are against another robot's interests, but we suppose they have the same task objective. In this context, we are interested in modeling relationships between robots that are on the same side and we do not intend to model fights between groups of robots.

3. Personality traits

One of the suggestions of the Theory of Evolution is that animals have emotions [18]. Moreover, these emotions are shared by the same species. Also, traits of personality (term that is used interchangeably with emotions) are important to the maintenance of objectives and collaboration [6]. Using these ideas, we define a way robots may react to their environment [7, 19].

In our problem, the robots are assumed, initially, to be homogeneous in configuration and capabilities (understood as the set of traits of personality available to each one of them). However, like ants in a colony, they should differ from each other in order to better perform a complex task. But we do not want to add complexity to our algorithms. In order to solve this dilemma, we make use of personality traits. Therefore, the algorithms are the same for every robot, but changing these numerical values (the "traits") will change the behaviour of individual robots. Although simple, this idea is very powerful and joined with reinforcement learning may lead to a heterogeneous population in which some robots are able to specialize in executing certain tasks but at the same time they may learn how to execute a different action if it is necessary.

Personality traits are represented by real numbers γ_i. They are used in order to represent the individual intentions when faced with changes in the environment. At each time t, the robot may take one action α_i chosen from a set of possible actions A. If we define X as the state of the robot and Y as the state of the environment, we may establish a function $f : X \times Y \times A \rightarrow X \times Y$, i.e., a function that maps the current states of a robot and the environment to different states through the execution of an action. The problem then becomes to determine a way to represent the states X and Y and relate them to the actions A. Clearly, if we increase the number of represented states and possible actions; the number of possibilities for the function f increases exponentially. This could be regarded as the main problem when a symbolic representation of the world is used.

The choice of actions is made considering the traits of personality a robot has and the payoffs related to each action at a given moment in time. By payoffs we mean the reward or penalty that may come from executing a specific action. For example, if by performing an action the robot gets closer to its objective, it receives a reward (a positive number that represents the possible success of the action); on the other hand, if as a consequence of an action the robot is damaged or increased its risk of being damaged, it receives a penalty (a negative number that represents the failure of the action). Notice that the payoffs may change according to a change in the representation of the environment. We may give a human example to explain that. In our diet we usually would not consider eating worms. However, if we were lost in a jungle, we would

do anything that would keep us alive. In the same way, a robot can change the values of its payoffs if the environment as it perceives it changes.

Let us make some definitions.

Definition 1 *For a player i, we define the personality traits to be*

$$\bar{\gamma}_i = [\gamma_1, \cdots, \gamma_n]^T \tag{1}$$

the reward functions defined for each one of the n personality traits are represented by the vector

$$\bar{\mathcal{E}}_i = [\mathcal{E}_1, \cdots, \mathcal{E}_n]^T \tag{2}$$

However, in this section we are going to drop the subscript i, since the algorithm will be explained for just one robot.

The reward functions $\mathcal{E}_j$ represent how well a personality trait contributed to the success of the robot. These functions are arbitrary and defined based on the problem under consideration [19]. When an action α_k is chosen to be executed, all personality traits $\bar{\gamma}$ are then updated.The effect of the action taken is evaluated using the equation

$$\mathcal{U}(s, \bar{\gamma}, \alpha_k) = h(\sum_{j=1}^{n} \gamma_j \mathcal{E}_j(s_t, \alpha_k, t)) \tag{3}$$

where the individual reward functions $\mathcal{E}_j$ are related to how beneficial for the robot the execution of action α_k in the presence of state s_t is according to each personality trait γ_j and, therefore, determines the reward and/or penalty related to the trait of personality. Function $h(\cdot)$ is a suitably defined function that wait the cost function inside the summation in a way particular to each problem under consideration. Notice that the reward functions $\mathcal{E}_j$ are some feedback from the environment. Since there could potentially be more robots acting on the environment, the reward that one particular robot gets depends indirectly on the actions taken by the other agents present. Furthermore we assume that the weights γ_j (i.e., the personality traits) are normalized, so

$$\sum_{j=1}^{n} \gamma_j = 1; \gamma_j \geq 0 \tag{4}$$

However notice that the personality traits vector $\bar{\gamma}$ is not a probability vector. In other words, γ_j is not the probability of the robot to take action j. Indeed, the dimension of the personality traits vector and the number of actions are, in general, not the same.

Equation (3) takes into account all the traits of personality γ_j and the environment as represented at time t (the current time step). Moreover, action α_k is the action under consideration and s_t is the state that the robot perceives the environment to be in at time t.

Furthermore, some procedure for evolving the personalities is necessary. The main purpose of that is to diminish the dilemma between stability and adaptability (plasticity) of learning.

The dynamics of the personality vector are described by the following general difference equation

$$\bar{\gamma}_{t+1} = \bar{\gamma}_t + \eta \bar{F}(\bar{\gamma}_t, \bar{\mathcal{E}}_t) \tag{5}$$

where the function $\bar{F}(\bar{\gamma}_t, \bar{\mathcal{E}}_t)$ depends on the application under consideration.

Equation (5) implies, because the utility function is a function of all personality traits, that each trait of personality influences the others (notice that the term $(\bar{F}(\bar{\gamma}_t, \bar{\mathcal{E}}_t))$ may include the reward functions for all the other personality traits). Therefore, changing a single trait will affect all the others. Furthermore, since the utilities include the plays of all other players (as explained above), the utilities and plays of other players also influence how the personality traits for one robot changes.

Equation (5) can be characterized by an ordinary differential equation. This is done in order to prove the convergence of the algorithm. We may rewrite (5) as

$$\frac{\bar{\gamma}_{t+1} - \bar{\gamma}_t}{\eta} = \bar{F}(\bar{\gamma}_t, \bar{\mathcal{E}}_t)$$

If we let $\eta \to 0$ we have

$$\dot{\bar{\gamma}} = \bar{F}(\bar{\gamma}, \bar{\mathcal{E}}) \tag{6}$$

Notice that this approximation may be very convenient in representing the process under study, but when we implement the algorithms we use (5).

4. Simulation Results

In order to demonstrate the ideas presented so far, we will now introduce three simulations that will incrementally become more complex and, together, will show how powerful the approach suggested is. We start, in section 4.1, with a description of the general framework for simulation. In section 4.2 we introduce a simulation of a zero-sum (matrix) game wherein no saddle point exists and therefore mixed strategies must be used [1]. A theoretical proof of convergence is given and we compare the theoretical optimal solution with the results obtained by the learning procedure depicted in this work.

In section 4.4 we make things a little bit more complex and model a robotic conflict using a non-zero-sum game. Moreover, this time, we will have more players (3 robots) and the reward will not only be the payoff table, but a combination of payoffs and goal achievement. There is no proof of convergence for this case and its existence is an open question. The results shown are based on heuristics.

Finally, in section 4.5 we introduce a situation in which the robots do not perceive the environment completely. Therefore, we will make use of potential fields for the representation of the environment and a still more complex learning procedure. This is also an open problem. However the results are very promising.

4.1. Simulation framework

All the simulations presented in this paper will follow a single framework. However, the meaning of some of the terms will change as we go forth in considering more complicated situations. The general framework is described in Algorithm 1 below.

For each of the simulations presented in this paper, the steps in Algorithm 1 could be expanded as needed. Specially, in the simulations shown in section 4.5, each of the steps above is implemented as a series of several items in the instantiated algorithm.

[1] For a gentle introduction to game theory, refer to [20]. For a more complete reference, [21] may be used.

Algorithm 1 General algorithm

1: Define, if necessary, the variables necessary to represent the environment and initialize them. This may take several steps of the algorithm.
2: Initialize learning rate η. This value is dependent on the problem under consideration.
3: Define how many personality traits are necessary for the problem under consideration and initialize them.
4: Define the payoffs for the game. These payoffs may be a matrix (or set of matrices) as in sections 4.2 and 4.4, or reward functions $\mathcal{E}_j$ representing the contribution of a trait of personality to the success of a mission combined with a matrix describing the state of the environment as in the simulation in section 4.5.
5: Play the game. The rules for the game will be introduced in each simulation. In general, we use an equation in the form of ((3).
6: Update the personality traits according to (26) and (27) in the case of the robots leaving the room and the robots tracking a target. In the case of the zero-sum game, the updating takes a slightly different form that will be described shortly.
7: Normalize (if necessary) personality traits γ_i for each robot according to (4).

4.2. A zero-sum game example

The game that we will analyze in this section is reported in [20] and is represented in Table 1. This is a zero-sum game, meaning that the payoffs for each player always add to zero. For example, if player A plays the strategy A1 and player B plays the strategy B1, player A gets rewarded 4 units while player B gets a penalty of 4; whereas, if player B decides to play strategy B2, then player B gets a reward of 4 units while player A gets penalized 4 units.

Table 1. Zero-sum game example

		Player B Strategies					
		B1	B2	B3	B4	B5	B6
Player A Strategies	A1	4	-4	3	2	-3	3
	A2	-1	-1	-2	0	0	4
	A3	-1	2	1	-1	2	-3

As may be seen in Table 1, there is no saddle point for the game, therefore there must be a mixed strategy set for both players. The optimal strategies, calculated using a linear programming solver such as the Simplex, for both players, are shown in Table 2.

Table 2. Optimal mixed strategies

Player	**Strategy**	**Optimal Frequency**
Player A	A1	24%
	A2	21%
	A3	55%
Player B	B1	0%
	B2	36%
	B3	0%
	B4	57%
	B5	0%
	B6	7%

4.2.1. Convergence

A general proof of convergence of strategies (or actions) to a Nash Equilibrium is virtually impossible to be obtained for equation (6) of section 3 or, equivalently, for the algorithm presented in section 4.1. Therefore, we now provide a proof of convergence for the special case of zero-sum games [2]. Our proof will follow closely the one found in [22]. However, one fundamental difference is made. Since we use personality traits, the strategies dynamics depend on them and additional considerations must be made. Therefore, the personality dynamics are introduced in order to derive the strategies dynamics.

For a zero-sum game, the utility functions for the two players involved would be:

$$\mathcal{U}_1(\bar{p}_1, \bar{p}_2) = \bar{p}_1^T M \bar{p}_2 \tag{7}$$

$$\mathcal{U}_2(\bar{p}_2, \bar{p}_1) = -\bar{p}_2^T M^T \bar{p}_1 \tag{8}$$

where $\bar{p_1} \in \mathbb{R}^n$ and $\bar{p_2} \in \mathbb{R}^m$ are arrays of probabilities where p_{i_n} is the probability that strategy n for player i be executed. The matrix $M \in \mathbb{R}^{n \times m}$ is the payoff matrix for both players. For the simulation at hand, matrix M is

$$M = \begin{bmatrix} 4 & -4 & 3 & 2 & -3 & 3 \\ -1 & -1 & -2 & 0 & 0 & 4 \\ -1 & 2 & 1 & -1 & 2 & -3 \end{bmatrix} \tag{9}$$

In order to proceed with our proof, we need to define some notation. This is done in the following definition.

Definition 2 *Consider $\bar{x} = [x_1, \cdots, x_n]$ where n represents the number of strategies*

- $\Delta(n)$ *denotes the simplex [10] in $\mathbb{R}^n$ (figure 1), i.e.,*

$$\Delta(n) = \{\bar{x} \in \mathbb{R}^n : \bar{x}_i \geq 0 \; \forall i = 1, \cdots, n; and \sum_{j=1}^{n} \bar{x}_i = 1\}$$

- $Int(\Delta(n))$ *is the set of interior points of a simplex [10], i.e.,*

$$int(\Delta(n)) = \{\bar{x} \in \Delta(n) : \bar{x}_i > 0 \; \forall i = 1, \cdots, n\}$$

- $bd(\Delta(n))$ *is the boundary of the simplex [10] (figure 1), i.e.,*

$$bd(\Delta(n)) = \{\bar{x} \in \Delta(n) : \bar{x} \notin int(\Delta(n))\}$$

- $\bar{v}_i \in bd(\Delta(n))$ *is the i^{th} vertex of the simplex $\Delta(n)$, i.e.,*

$$\bar{v}_i = \{\bar{x} \in \Delta(n) : \bar{x}_i = 1 \; and \; \bar{x}_j = 0 \; \forall j \neq i\}$$

[2] In the particular case of zero-sum games, the convergence is to a Nash equilibrium.

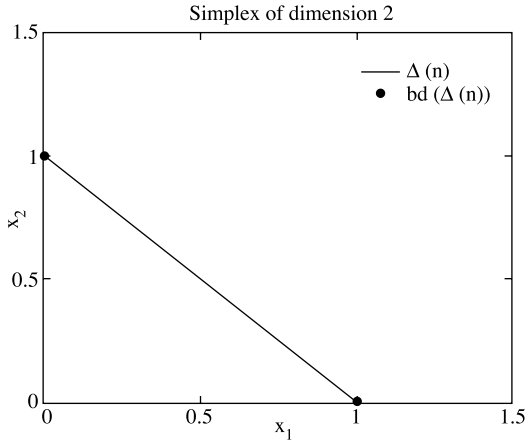

Figure 1. Simplex of a player with two strategies

Now define the best response mappings as

$$\bar{\beta}_1(\bar{p}_2) = \arg \max_{\bar{p}_1 \in \Delta(n)} \mathcal{U}_1(\bar{p}_1, \bar{p}_2) \tag{10}$$

$$\bar{\beta}_2(\bar{p}_1) = \arg \max_{\bar{p}_2 \in \Delta(m)} \mathcal{U}_2(\bar{p}_2, \bar{p}_1) \tag{11}$$

The utilities in (7) and (8) are implementations of the utility function (3). We also need to define the reward functions related to each personality trait (the functions $\mathcal{E}_j$) and the function used to update the personality traits (function $\bar{F}(\bar{\gamma}, \bar{\mathcal{E}})$ used in (6)) for each of the two players. Note that the arguments for the utility functions are apparently different from the parameters found in (3), however, the parameters are embedded in the definitions of $\bar{p}_1$ and $\bar{p}_2$, for they are dependent on $\bar{\gamma}_1$ and $\bar{\gamma}_2$ respectively (13). Function $h(\cdot)$ is the identity.

Let us now define the concept of empirical frequency (expectation of the opponent executing each of its actions). The empirical frequency $\bar{q}_i$ is calculated as the running average [22] of the observed actions of the opponent (recall that we have access to the action the opponent has played at each time step)

$$\begin{aligned} \bar{q}_1(k) &= \bar{q}_1(k-1) + \frac{1}{k}(\bar{v}_{a_1(k-1)} - \bar{q}_1(k-1)) \\ \bar{q}_2(k) &= \bar{q}_2(k-1) + \frac{1}{k}(\bar{v}_{a_2(k-1)} - \bar{q}_2(k-1)) \end{aligned} \tag{12}$$

where $a_i(k-1)$ is the action executed by player i at time $k-1$ and $\bar{v}_i$ is a vertex of the simplex as defined in definition 2. We may assume that as $k \to \infty$, $\bar{q}_i(k) \to \bar{p}_i$. Therefore, in the proof, we are going to use both terms interchangeably.

Let us define a mapping from the personality space to the strategy space $d : \mathbb{R}^r \to \Delta(n)$ as

$$\begin{aligned} \bar{q}_1 &= A_1 \bar{\gamma}_1 \\ \bar{q}_2 &= A_2 \bar{\gamma}_2 \end{aligned} \tag{13}$$

Notice that A transforms from personalities to likelihood of actions. We can now define the reward

functions for each personality trait defined as (for each player)

$$\begin{aligned} \bar{\mathcal{E}}_1 &= {A_1}^T(A_1{A_1}^T)^{-1}\bar{\beta}_1(\bar{q}_2) \quad (14)\\ \bar{\mathcal{E}}_2 &= {A_2}^T(A_2{A_2}^T)^{-1}\bar{\beta}_2(\bar{q}_1) \quad (15)\end{aligned}$$

where, $A_1 \in \mathbb{R}^{n\times r_1}$ and $A_2 \in \mathbb{R}^{m\times r_2}$. These reward functions make use of the pseudo-inverse and are used in the convergence proof. In our example (with matrix M defined in (9)), $n = 3$ strategies and $m = 6$ strategies. Therefore, we select $r_1 = 5$ personality traits and $r_2 = 10$ personality traits. Notice that according to (14) and (15) $r_1 \geq n$ and $r_2 \geq m$. Moreover, $rank(A_1) = n$ and $rank(A_2) = m$. Other than those, no assumption is taken. Furthermore, the functions used to update the personality traits (the $\bar{F}(\bar{\gamma}, \bar{\mathcal{E}})$ in (6)) are

$$\begin{aligned} \bar{F}_1(\bar{\gamma}_1, \bar{\mathcal{E}}_1) &= \bar{\mathcal{E}}_1 - \bar{\gamma}_1\\ \bar{F}_2(\bar{\gamma}_2, \bar{\mathcal{E}}_2) &= \bar{\mathcal{E}}_2 - \bar{\gamma}_2 \quad (16)\end{aligned}$$

The resulting personality dynamics are

$$\begin{aligned} \dot{\bar{\gamma}}_1 &= {A_1}^T(A_1{A_1}^T)^{-1}\bar{\beta}_1(\bar{q}_2) - \bar{\gamma}_1\\ \dot{\bar{\gamma}}_2 &= {A_2}^T(A_2{A_2}^T)^{-1}\bar{\beta}_2(\bar{q}_1) - \bar{\gamma}_2 \quad (17)\end{aligned}$$

In this simulation we relaxed the condition of normalization presented in (4). However, this is just for simplicity of calculations. Should we have wanted to do so, this could be easily implemented. Matrices A_1 and A_2 used were

$$A_1 = \begin{bmatrix} 0.3267 & 0.5071 & 0.7707 & 0.0478 & 0.3606\\ 0.5406 & 0.7828 & 0.9703 & 0.1291 & 0.4767\\ 0.1427 & 0.2456 & 0.3197 & 0.9082 & 0.2506 \end{bmatrix} \quad (18)$$

and

$$A_2 = \begin{bmatrix} 0.8686 & 0.6813 & 0.0693 & 0.2760 & 0.5695 & 0.5676 & 0.6390 & 0.6081 & 0.1034 & 0.1500\\ 0.6264 & 0.6658 & 0.8529 & 0.3685 & 0.1593 & 0.9805 & 0.6690 & 0.1760 & 0.1573 & 0.3844\\ 0.2412 & 0.1347 & 0.1803 & 0.0129 & 0.5944 & 0.7918 & 0.7721 & 0.0020 & 0.4075 & 0.3111\\ 0.9781 & 0.0225 & 0.0324 & 0.8892 & 0.3311 & 0.1526 & 0.3798 & 0.7902 & 0.4078 & 0.1685\\ 0.6405 & 0.2622 & 0.7339 & 0.8660 & 0.6586 & 0.8330 & 0.4416 & 0.5136 & 0.0527 & 0.8966\\ 0.2298 & 0.1165 & 0.5365 & 0.2542 & 0.8636 & 0.1919 & 0.4831 & 0.2132 & 0.9418 & 0.3227 \end{bmatrix} \quad (19)$$

The values of matrices in 18 and 19 were generated randomly. The point is that if the matrices satisfy the conditions listed above, the algorithm will converge. More importantly, if A_1 and A_2 are the identity matrices of necessary dimensions, the algorithm reduces to fictitious play and convergence is still guaranteed [13].

Using these definitions, we find that the strategy dynamics for player 1 is

$$\begin{aligned} \dot{\bar{q}}_1(t) &= A_1\dot{\bar{\gamma}}_1(t)\\ \dot{\bar{q}}_1(t) &= A_1({A_1}^T(A_1{A_1}^T)^{-1}\bar{\beta}_1(\bar{q}_2(t)) - \bar{\gamma}_1(t))\\ \dot{\bar{q}}_1(t) &= \bar{\beta}_1(\bar{q}_2(t)) - \bar{q}_1(t) \quad (20)\end{aligned}$$

In the same way, the strategy dynamics for player 2 is

$$\dot{\bar{q}}_2(t) \quad = \quad \bar{\beta}_2(\bar{q}_1(t)) - \bar{q}_2(t) \tag{21}$$

We now define a function that measures the maximal possible reward for the players

$$V_1(\bar{q}_1, \bar{q}_2) = \max_{\bar{x}\in\Delta(n)} \mathcal{U}_1(\bar{x}, \bar{q}_2) - \mathcal{U}_1(\bar{q}_1, \bar{q}_2) \tag{22}$$

where $\max_{x\in\Delta(n)} \mathcal{U}_1(\bar{x}, \bar{q}_2)$ is the best "strategy" [3] that may be used by player 1. Using (10), we know that

$$\max_{x\in\Delta(n)} \mathcal{U}(\bar{x}, \bar{q}_2) = (\bar{\beta}_1(\bar{q}_2))^T M \bar{q}_2 \tag{23}$$

Therefore, from (22), using the definitions of utility functions in (7) and (8) together with the definition of best response mappings in (10) and (11), and collecting terms one gets

$$V_1(\bar{q}_1, \bar{q}_2) \quad = \quad (\bar{\beta}_1(\bar{q}_2) - \bar{q}_1)^T M \bar{q}_2 \tag{24}$$

In the same way

$$V_2(\bar{q}_2, \bar{q}_1) = -(\bar{\beta}_2(\bar{q}_1) - \bar{q}_2)^T M^T \bar{q}_1 \tag{25}$$

Finally, we may say that $V_1(\bar{q}_1, \bar{q}_2) \geq 0$ and $V_2(\bar{q}_2, \bar{q}_1) \geq 0$ with equality if and only if $\bar{q}_1 = \bar{\beta}_1(\bar{q}_2)$ and $\bar{q}_2 = \bar{\beta}_2(\bar{q}_1)$.

Now we prove that the learning procedure will converge to the optimal solution. We start by the following Lemma.

Lemma 1 *Define* $\tilde{V}_1(t) = V_1(\bar{q}_1(t), \bar{q}_2(t))$ *and* $\tilde{V}_2(t) = V_2(\bar{q}_2(t), \bar{q}_1(t))$. *Then* $\dot{\tilde{V}}_1(t) = -\tilde{V}_1(t) + \dot{\bar{q}}_1{}^T M \dot{\bar{q}}_2$ *and* $\dot{\tilde{V}}_2(t) = -\tilde{V}_2(t) - \dot{\bar{q}}_1{}^T M^T \dot{\bar{q}}_2$.

Proof By definition (22)

$$\begin{aligned}
\dot{\tilde{V}}_1(t) &= \frac{d}{dt}[\max_{\bar{x}\in\Delta(n)} \mathcal{U}_1(\bar{x}, \bar{q}_2(t)) - \mathcal{U}_1(\bar{q}_1(t), \bar{q}_2(t))] \\
&= \frac{d}{dt}[\max_{\bar{x}\in\Delta(n)} \mathcal{U}_1(\bar{x}, \bar{q}_2(t))] - \frac{d}{dt}[\bar{q}_1^T(t) M \bar{q}_2(t)] \\
&= \nabla_{q_2}[\max_{\bar{x}\in\Delta(n)} \mathcal{U}_1(\bar{x}, \bar{q}_2(t))]\dot{\bar{q}}_2(t) - \dot{\bar{q}}_1^T(t) M \bar{q}_2(t) - \bar{q}_1^T(t) M \dot{\bar{q}}_2(t)
\end{aligned}$$

We now use the fact that [23]([22], Lemma 3.2)

$$\nabla_{q_2} \max_{\bar{x}\in\Delta(n)} \mathcal{U}_1(\bar{x}, \bar{q}_2(t)) = \nabla_{q_2} \max_{\bar{x}\in\Delta(n)} [\bar{x}^T M \bar{q}_2(t)] = \bar{\beta}_1^T(\bar{q}_2(t)) M$$

that yields (using (20) and (24))

$$\begin{aligned}
\dot{\tilde{V}}_1(t) &= \bar{\beta}_1^T(\bar{q}_2(t)) M \dot{\bar{q}}_2(t) - (\bar{\beta}_1(\bar{q}_2(t)) - \bar{q}_1(t))^T M \bar{q}_2(t) - \bar{q}_1^T(t) M \dot{\bar{q}}_2(t) \\
&= -(\bar{\beta}_1(\bar{q}_2(t)) - \bar{q}_1(t))^T M \bar{q}_2(t) + (\bar{\beta}_1(\bar{q}_2(t)) - \bar{q}_1(t))^T M \dot{\bar{q}}_2(t) \\
&= -\tilde{V}_1(t) + \dot{\bar{q}}_1^T(t) M \dot{\bar{q}}_2(t)
\end{aligned}$$

[3] For a discussion on pure and mixed strategies, refer to [10]

Therefore $\dot{\tilde{V}}_1(t) = -\tilde{V}_1(t) + \dot{\bar{q}}_1^T(t) M \dot{\bar{q}}_2(t)$. A similar derivation may be used for $\tilde{V}_2(t)$, yielding $\tilde{V}_2(t) = -\tilde{V}_2(t) - \dot{\bar{q}}_2^T(t) M^T \dot{\bar{q}}_1(t)$. □

And, finally, we enunciate the convergence theorem

Theorem 1 *The solutions of the system of differential equations ((20) and (21)) satisfy* $\lim_{t\to\infty}(\bar{q}_1(t) - \bar{\beta}_1(\bar{q}_2(t))) = 0$ *and* $\lim_{t\to\infty}(\bar{q}_2(t) - \bar{\beta}_2(\bar{q}_1(t))) = 0$.

Proof We know from Lemma 1 that $\dot{\tilde{V}}_1(t) = -\tilde{V}_1(t) + \dot{\bar{q}}_1^T(t) M \dot{\bar{q}}_2(t)$ and $\dot{\tilde{V}}_1(t) = -\tilde{V}_2(t) - \dot{\bar{q}}_2^T(t) M \dot{\bar{q}}_1(t)$. Define the function $V_{12}(t) = \tilde{V}_1(t) + \tilde{V}_2(t)$. Taking its derivative, we have

$$\begin{aligned} \dot{V}_{12}(t) &= \dot{\tilde{V}}_1(t) + \dot{\tilde{V}}_2(t) \\ &= -\tilde{V}_1(t) - \tilde{V}_2(t) \end{aligned}$$

Since $\tilde{V}_1(t) \geq 0$ and $\tilde{V}_2(t) \geq 0$ with equality only at the equilibrium point of (20) and (21), $\tilde{V}_{12}(t)$ is a Lyapunov function and the theorem follows the arguments. □

4.2.2. Simulation results

We now present the results for the problem presented in the matrix in (9). Suppose that both players are initialized with personality traits set to random values (in the interval $[0, 1]$). This means that both players start with a random probability of playing each strategy that is different from the optimal showed on Table 2. The question we want to answer is: if we use personalities as described above, will the players learn to play the best mixed strategy? If not, will there be any improvement over time?

Algorithm 2 is followed by both players. As described in (18) and (19), we have created 5 personality traits for player 1 and 10 personality traits for player 2. Observe that this is not necessary and we could have used a much larger number of personalities (or, on the other hand, a smaller number greater or equal to the number of strategies) to characterize our player.

The utility functions are defined in (7) and (8). If we discretize the equation of dynamics for the personality traits ((20) and (21)) we end up with equations of the form found in (5) where, again, the value η may be called the learning rate and is set to a positive number and functions $\bar{F}(\bar{\gamma}, \bar{\mathcal{E}})$ are defined as in (16).

We then ran the simulation for 5000 iterations and collected the final probabilities of using each one of the 3 strategies at player A's disposal and each one of the 6 strategies at player B's disposal as well as the value that player B obtained in each simulation (the value obtained by player A is just the negative of the one obtained by player B). Such results are summarized in Table 3. One may notice that, indeed, the procedure made the personalities converge to their optimal values. Furthermore, we observe that the disadvantageous strategies B1 and B3 were almost never used. Moreover, strategy B5, which is dominated by strategy B2, was never used. The learning procedure even caught the subtlety that B2 dominates B5. Furthermore, the average payoff for player B was 0.0710, very close to the theoretical value of the game, 0.07 [20].

The importance of this example resides in the fact that the robots are able to learn from experience when the task is represented in terms of a game. Notice that for a zero-sum game, the approach reduces to

Algorithm 2 Zero-sum game

1: $\eta \leftarrow 0.1$
2: $t \leftarrow 0.01$
3: Set the number of personality traits for player A equal to 5 and for player B equal to 10.
4: Initialize the personality traits $\bar{\gamma}_A$ and $\bar{\gamma}_B$ randomly.
5: Define the payoffs for the game to be according to the matrix in (9). Also, define the mappings from the personality traits spaces to the strategies (or actions) spaces according to the matrices in (18) and (19).
6: Player A initialize the empirical frequency of player B, p_2, as 0. Player B initialize the empirical frequency of player A, p_1, as 0.
7: **for** $i = 1$ to 5000 **do**
8: Player A calculates the strategy to play according to its probability distribution $\bar{q}_1 = A_1\bar{\gamma}$. The action chosen is called a.
9: Player B calculates the strategy to play according to its probability distribution $\bar{q}_2 = A_2\bar{\gamma}$. The action chosen is called b.
10: Both players play their actions.
11: The payoff for player A is M_{ab} and the payoff for player B is $-M_{ab}$.
12: Update personality traits as described in (5).
13: Player A updates empirical frequency of player B according to (12).
14: Player B updates empirical frequency of player A according to (12).
15: Record the payoff for player B ($-M_{ab}$).
16: **end for**

Table 3. Experimental results obtained for both players

Player	Strategy	Optimal Frequency	Experimental Frequency
Player A	A1	24%	22.04%
	A2	21%	23.93%
	A3	55%	54.03%
Player B	B1	0%	1.05%
	B2	36%	36.10%
	B3	0%	0.02%
	B4	57%	56.39%
	B5	0%	0%
	B6	7%	6.44%

fictitious play. Therefore, convergence to the Nash Equilibrium will always occur. This is not the case for the other simulations presented in this paper.

4.3. Implementation for next sections

In the next two simulations we are going to use the following implementation.

Going back to (5), we have to define how the function $\bar{F}(\bar{\gamma}, \bar{\mathcal{E}})_t$ will behave. Define $\Delta\mathcal{E}_i(t) = \mathcal{E}_i(s_t, \alpha_j, t) - \mathcal{E}_i(s_{t-1}, \alpha_k, t-1)$ where α_j is the action taken at time t and α_k was the action taken at time $t-1$, we define the step as

$$\Delta\gamma_i(t) = \frac{\Delta\mathcal{E}_i(t)}{\sum_{j=1}^{n} \Delta\mathcal{E}_j(t)} \tag{26}$$

where n is the number of personality traits of a robot. Now we define the adaptation law as

$$\gamma_i(t) = \gamma_i(t-1) + \eta\Delta\gamma_i(t),\ 0 < \eta < 1 \tag{27}$$

Equations (26) and (27) imply, because of the presence of all personality traits in the denominator, that each trait of personality influences the others. Therefore, changing a singular trait will affect the way all the others work. Furthermore, the convergence of (27) is highly dependent on the value of the learning rate η: if this is small, the convergence is too slow and the robots take too long to adapt to new situations; if it is too high, the system oscillates a lot around some value before it converges and when it does, it has a large probability to go to a local maximum (minimum). In our simulations, we use a small value for this term so that we achieve a smooth convergence. Equation (27) is in the same form used in the reinforcement learning literature. More details may be found in [24]. Finally, notice that nothing can be said about the convergence of the algorithm at this point, for the coefficient of $\bar{\gamma}_t$ is not known.

When the state is recognized, the action is chosen according to the formula, known as the randomized strategy [24], which is useful for leading the robot to "explore" new actions and not just "exploit" learnt sequence of actions. The next equation demonstrates the randomized strategy.

$$P(\alpha_i|s) = \frac{k^{\mathcal{U}(s,\bar{\gamma},\alpha_i)/T}}{\sum_{j=1}^{m} k^{\mathcal{U}(s,\bar{\gamma},\alpha_j)/T}} \tag{28}$$

In (28), P is the probability that action α_i will be executed when the state is s in the presence of the personality traits $\bar{\gamma}$, where $\bar{\gamma} = [\gamma_1, \cdots, \gamma_n]$ is a vector of personality traits (as introduced in section 3). The term k is a coefficient that defines how often the robot will explore new solutions or exploit the ones it already knows as better ones. When k increases, the probability that the robot will explore new choices decreases and vice-versa. T is a temperature parameter inspired in *Boltzmann* theory of statistical mechanics. It is desired that over time, T decreases to diminish exploration [24]. The utility function $\mathcal{U}(.)$ is related to the action under consideration and the current state. $\mathcal{U}(.)$ is the long term expected reward and not just the instantaneous one. This means that the decisions the robot takes are based in the expectation to solve the problem (find a target or perform a predefined task) and not just in the instantaneous reward. This difference is implemented in the personalities, since robots learn over time, they will develop the capability to "predict" the payoffs of their actions. Moreover, n is the number of personality traits. And finally m is the number of possible actions that a robot may perform (in our present case it is the number of different strategies available for the robot). In this paper, in all simulations we will use $k = e$ (i.e., $exp(1) = 2.7183$) and $T = 1$. $T = 1$ means that we do not reduce exploration as time goes by. $k = e$ means that the robots have a preference for exploitation of already learnt strategies but are also open to exploration [24].

4.4. Robots leaving a room

Our second simulation is similar to the one introduced and reported in [19]. The setup is the following. Three robots of 1 unit in diameter are located in a room with dimensions 8×8 (corners at $(0,0)$, $(0,8)$, $(8,8)$ and $(8,0)$). There is a door centered at $(3,8)$ and with dimensions so that just one robot may pass. Inside the room, there are three robots located at positions $(3+3\sqrt{2}, 8-3\sqrt{2})$, $(3,6)$ and $(3-3\sqrt{2}, 8-3\sqrt{2})$, i.e., at a distance 6 units from the door (figure 2). It is assumed that one robot knows the positions of the other two without any noise, and it is also assumed that the robots only move in a straight line from their initial position toward the center of the door with fixed speed, 1 unit/s. The problem may be described as a game shown in Table 4, which has the payoffs for player A. The values X and Y in the table must follow the rule

$$X \in \mathbb{Z}^+ \; and \; Y \in \mathbb{Z}^- \tag{29}$$

where the values for X and Y will depend on how the designer chooses to represent the environment. If

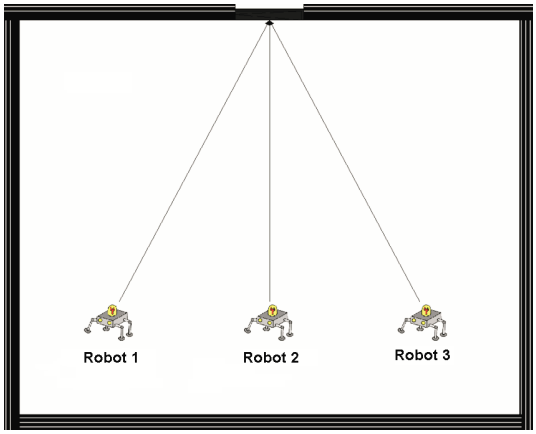

Figure 2. Artistic depiction of the problem of robots leaving a room

Table 4. Modelling of a game between two robots trying to leave a room

		Player B Strategies	
		Walk	Wait
Player A Strategies	Walk	-1	X
	Wait	Y	0

the designer wants to enhance the action "Walk", then set $|X| > |Y|$. On the other hand, if the designer wants to enhance the action "Wait", then set $|Y| > |X|$. And finally, if both are considered to be at the same level, set $|X| = |Y|$.

Before we state the algorithm, we need to make some definitions.

Definition 3 *Definitions for algorithm 3.*

i. The payoffs for the robots 1 and 3 in Figure 2 are according to the matrix

$$M_1 = \begin{bmatrix} -1 & 1 \\ -1 & 0 \end{bmatrix} \tag{30}$$

i.e., where $|X| = 1$ and $|Y| = 1$ (see Table 4 and equation (29)). For player 2, the payoff table is

$$M_2 = \begin{bmatrix} -1 & 3 \\ -1 & 0 \end{bmatrix} \tag{31}$$

where $X = 3$ and $Y = -1$ (see Table 4 and equation (29)). The values are different such that robot 2's expected reward is greater than its expected penalty.

ii. The probability for a robot to move is given by

$$P(Walk) = \frac{e^{\gamma_1}}{e^{\gamma_1} + e^{\gamma_2}} \tag{32}$$

Algorithm 3 Robots Leaving the Room

1: Each robot will have two personality traits, initialized as $\gamma_i = \frac{1}{2},\ i = 1, 2$, which define which strategy (Walk or Wait in Table 4) the robot will play.
2: Define the payoffs for the robots 1 and 3 according to (30) and the payoffs for robot 2 according to (31).
3: $Robots \Leftarrow [1, 2, 3]$.
4: **while** there are robots in the room **do**
5: For each robot calculate the probability to move according to the personality traits. The probability to move is given by (32), i.e., $A_l \in \{Walk, Wait\}$ (where l is the robot's id).
6: **for** $l \in Robots$ **do**
7: **if** no other robot is the the room **then**
8: There is no conflict. Set action to "Walk", i.e., $A_l \Leftarrow Walk$.
9: **else**
10: **if** $l = 1$ or $l = 3$, $M \Leftarrow M_1$ (30), otherwise $M \Leftarrow M_2$ (31)
11: **for** $j \in Robots,\ j \neq l$ **do**
12: $F(\gamma_{A_l}, \mathcal{E}_{A_l}) \Leftarrow M(A_l, A_j)$ {Payoff for robot l playing against robot j.}
13: Update the personality trait related to the action chosen according to the equation $\gamma_{A_l}(t) \Leftarrow \gamma_{A_l}(t-1) + \eta F(\gamma_{A_l}, \mathcal{E}_{A_l})$.
14: **end for**
15: Normalize all personality traits γ_i so that $\sum_{k=1}^{2} \gamma_k = 1; \gamma_k \geq 0$.
16: **end if**
17: Add action A_l to list of actions L_l taken so far.
18: **if** action is to walk and there is no collision **then** robot l moves **else** robot l keeps its current position for one time step.
19: **if** robot l reached door **then** remove l from list $Robots$
20: **end for**
21: **end while**

The results were obtained after 100 repetitions of the game with a learning rate $\eta = 0.01$. First of all, one of the robots on the sides (robots 1 and 3) converged to a purely "cooperative" robot, i.e., its personality trait for waiting for the others became 1; whereas the other robot on the side converged to a purely "competitive" robot, i.e., its personality trait for always walking became 1. Second of all, the robot in the middle chooses its actions on a $50 - 50$ basis. As result of all this, the average of the 100 games is 10.04s to leave the room and the standard deviation is 1.82s. Also, 24 out of the 100 repetitions obtained the best solution of 8s [19].

One may notice that the robots did not work only for their own advantage. Table 4 shows that the strategy "Wait" is dominated by the strategy "Walk". However, behaving the way they did made the overall result much better for the group. This is one of the interesting results that will be exploited in the next simulation. Here we see the spontaneous emergence of altruistic behaviour that enhances the performance of the group. The emergence of altruism is due to the calculations done in steps 13 and 15 of the algorithm. Notice that the matrices in (30) and (31) do not have a positive payoff for the strategy "Wait". However, since the traits that determine the execution of the actions are normalized (step 15), the negative payoffs that the strategy "Wait" gets combined with the negative payoffs of the collisions when strategy "Walk" is chosen will drive one of the robots to be altruistic.

4.5. Tracking a target

We now make use of all the ideas presented in this paper, and define a more complex and challenging simulation mission to be accomplished by several robots working together. We set up the simulation environment as shown in Figure 3. In this figure, we depict a target (a tank) and several robots that

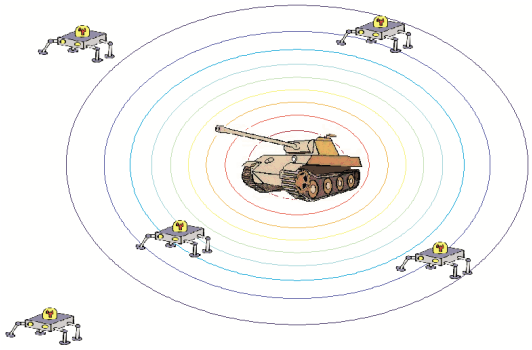

Figure 3. Artistic depiction of the simulation environment.

are moving around it. Their objectives are to find the target and go back to the base. In our simulation environment, the position of the target changes from simulation to simulation and the robots perceive the environment as potential fields (Gaussian potential fields). Each single robot is able to identify the target potential field, the other robots fields and the bases field. No noise is added to the readings and some delay is possible in the measurements. We also assume that the low-level dynamics of the robots and the control loops necessary to stabilize them are already implemented.

Each robot has three traits of personality: "courage" (γ_1), "fear" (γ_2) and "cooperation" (γ_3), which influence which action the robot will take. For example, a courageous robot may pursue the gradient of the target, while a cooperative and fearful one may tend to huddle together with other robots in order to look for the target as a group. Again, these behaviours are derived from our assumptions on the definition of the "emotions" of the robots.

For this simulation, the environment is supposed to be in only two states: θ_1, meaning high risk for the robot (of being shot) and θ_2, which means that the robot is in a low risk of being shot. The decision about which state the robot is in is psychological, i.e., it depends on the values of traits of personality of each robot. In this way, if a robot is "courageous" high risk has a different meaning compared to a "fearful" robot.

Let $\sigma(.)$ be a function determining the threshold in separating states θ_1 and θ_2. Let also γ_1 be the trait "courage", γ_2 be "fear" and γ_3 be "cooperation". Define F_{Max} as the maximum potential field found so far. We then define the probability for the robot to identify the environment as state θ_1 (high risk) as

$$P(\theta_1|s) = \frac{|F_T|}{|F_{Max}|} - \sigma(\gamma_1, \gamma_2, \gamma_3) \tag{33}$$

Since the traits of personality are normalized (as explained in the last sections), we chose the threshold function to be

$$\sigma(\gamma_1, \gamma_2, \gamma_3) = \gamma_1 - \gamma_2 - \gamma_3 \tag{34}$$

Therefore, if the trait of personality γ_1, "courage", is dominant, the probability the robot will identify the environment as being "high risk" will decrease. On the other hand, since $P(\theta_2|s) = 1 - P(\theta_1|s)$, the probability increases when "fear" (γ_2) and "cooperation" (γ_3) are dominant. Notice that $P(.)$ could be out of the interval $[0, 1]$, if that happens we simply truncate it.

In the same way, only two actions are possible. We shall call them α_1, which means to follow an uphill approach (getting closer to the dangerous target); and α_2, which means to follow a downhill path (according

Table 5. Utility payoffs for states

Utility Payoffs		States	
		θ_1	θ_2
Actions	α_1	-1	5
	α_2	4	-2

to danger). Table 5 describes the payoffs related to each decision when the robot identifies the environment to be in each specific state. The values in Table 5 are empirical and by choosing different payoffs the robots would end up with different behaviours. Also notice that the table is not exactly a payoff table as we had in the previous examples; in this case we do not have a conflict among the robots. The numbers in the table mean that when the robot perceives the state to be in the "high risk" state θ_1, it is more "profitable" to execute action α_2 (downhill path) and when the robot finds itself in the low risk state θ_2, the robot would prefer to execute action α_1 (uphill path). Later on (in equation (35)), we will see that the choice is not so straightforward, but in general the rules just explained will be applied.

After the robot identifies the target, it gets back to the base with its estimation of the target location. The closer the robot gets to the target, the greater the danger of being shot (at each time step we divide the potential field where the robot is by the maximum value of the field - the position of the target - and according to this number, randomly shoot at the robot simulating an action taken by the enemy). When the robot is shot, we assume that it is still operational, but has to go back to the base in order to avoid malfunctioning. Actually, since we may have a large number of robots, this assumption is not necessary, but by making use of it, we simplify our simulation environment. When the robot is shot, we artificially increase its "fear" trait of personality in order to avoid being shot in the future. The task "get back to base" is hardwired in this approach and after the robot identifies the target it just follows the track back to safety. Notice that this behaviour is artificial and not desired, for the robot must be able to help other robots in need even if it is on its way back to the base. However, we do not implement this feature for the sake of simplicity.

The traits of personality are defined as follows:

1. Courage (γ_1) the robot goes in the direction of danger, i.e., in the direction of the increasing potential field, therefore, this trait makes it more likely for the robot to identify the environment as in the "low risk" state (state θ_2 in Table 5).

2. Fear (γ_2) the robot goes in the opposite direction of danger, i.e., in the direction of the decreasing potential field, therefore, this trait makes it more likely for the robot to identify the environment as in the "high risk" state (state θ_1).

3. Cooperation (γ_3) robots tend to huddle together in order to decrease the possibility of being shot. This trait makes the robots work together.

The behaviour in 3 is explained by the assumption in the simulation that the chance of the robot being shot is inversely proportional to the number of robots huddled together. This is not a deliberate hypothesis; in fact the same assumption has been taken when studying the formation of patterns of animals in the wild (flock formation, fish schooling, etc.) [25].

To choose an action, we use the value function $\mathcal{U} : X \times A \rightarrow R$ in (35) that maps the state of the environment and the action under consideration to a reward. In the case of game theory we need to

calculate the expected value of the value function. Therefore, define $J(s_t, \bar{\gamma}, \alpha_i) = \sum_{j=1}^{3} \gamma_j \mathcal{E}_j(s_t, \alpha_i, t)$, i.e., the summation in (3). Now, define $\mathcal{U}(s_t, \bar{\gamma}, \alpha_i) = \mathbb{E}\{J(s_t, \bar{\gamma}, \alpha_i)|(s, \alpha_i)\}$ as the expected value for the payoff for all possible actions α_i. We can think of this as a game against nature [20], in which the environment is supposed to play with a mixed strategy $P(\theta_i|s)$. Therefore, the expected outcome of the game in table 5 is

$$\begin{aligned} \mathcal{U}(s_t, \bar{\gamma}, \alpha_1) = \mathbb{E}\{J(s_t, \bar{\gamma}, \alpha_1)|(s_t, \alpha_1)\} &= [-1(P(\theta_1|s) + 5(P(\theta_2|s)]J(s, \bar{\gamma}, \alpha_1) \\ \mathcal{U}(s_t, \bar{\gamma}, \alpha_2) = \mathbb{E}\{J(s_t, \bar{\gamma}, \alpha_2))|(s_t, \alpha_2)\} &= [4(P(\theta_1|s) - 2(P(\theta_2|s)]J(s_t, \bar{\gamma}, \alpha_2) \end{aligned} \quad (35)$$

Equation (35) is the application of game theory expectation calculation to the framework introduced in section 3. Actually, this equation is just the implementation of (3) in terms of game theory, where $h(\cdot)$ is the expectation function.

Then, the action is chosen randomly based on the probability ((28), where $k = e$ $(exp(1) = 2.7183)$, $T = 1$)

$$P(\alpha_i) = \frac{e^{\mathcal{U}(s, \bar{\gamma}, \alpha_i)}}{\sum_{j=1}^{2} e^{\mathcal{U}(s, \bar{\gamma}, \alpha_j)}} \quad (36)$$

Before we state the algorithm, we need to introduce some definitions:

Definition 4 *Definitions for algorithm 4.*

i. The target is identified by a Gaussian field. If (X_T, Y_T) designates the position of the target, let

$$T(x, y) = Ke^{-\frac{1}{2}\frac{(x-X_T)^2}{\sigma^2}} e^{-\frac{1}{2}\frac{(y-Y_T)^2}{\sigma^2}} \quad (37)$$

be the Gaussian field irradiated by it. σ is the standard deviation of the field and K is a term to scale the sensitivity of the robots.

ii. The robots also irradiate Gaussian fields. If (X_R, Y_R) is the position of a robot, let

$$R(x, y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(x-X_R)^2}{\sigma^2}} e^{-\frac{1}{2}\frac{(y-Y_R)^2}{\sigma^2}} \quad (38)$$

be the Gaussian field around it. σ is the standard deviation of the field.

iii. The uphill unit vector for robot i located at (x_i, y_i) is

$$\vec{u}_i = \frac{\nabla T(x_i, y_i) + \sum_{j \neq i} \nabla R_j(x_i, y_i)}{|\nabla T(x_i, y_i) + \sum_{j \neq i} \nabla R_j(x_i, y_i)|} \quad (39)$$

iv. The downhill unit vector for robot i located at (x_i, y_i) is

$$\vec{d}_i = -\vec{u}_i \quad (40)$$

v. The probability for the robot identifying that it is in state θ_1, high risk, is

$$P(\theta_1|s_t) = \frac{|\nabla T(x_i, y_i) - \sum_{j \neq i} \nabla R_j(x_i, y_i)|}{|F_{Max}|} - \gamma_1 + \gamma_2 + \gamma_3 \tag{41}$$

where $|F_{Max}|$ is the maximum field found so far for each robot. Accordingly, the probability to be in state θ_2, low risk, is

$$P(\theta_2|s_t) = 1 - P(\theta_1|s_t) \tag{42}$$

vi. The probability of executing action α_1 is

$$P(\alpha_1) = \frac{e^{\mathcal{U}(s,\bar{\gamma},\alpha_1)}}{e^{\mathcal{U}(s,\bar{\gamma},\alpha_1)} + e^{\mathcal{U}(s,\bar{\gamma},\alpha_2)}} \tag{43}$$

Accordingly, $P(\alpha_2) = 1 - P(\alpha_1)$. Equation (43) is (28), where $k = e$, $T = 1$ and $n = 2$.

vii. The personality traits are updated using the adaptation law

$$\gamma_i(t) = \gamma_i(t-1) + \eta \Delta\gamma_i(t) \tag{44}$$

where

$$\Delta\gamma_i(t) = \frac{\Delta\mathcal{E}_i(t)}{\sum_{j=1}^{3} \mathcal{E}_j(t)} \tag{45}$$

viii. The probability of a robot being shot is

$$P(shot) = \frac{|\nabla T(x_i, y_i) - \sum_{j \neq i} \nabla R_j(x_i, y_i)|}{\max(|T(x_j, y_j)|)} \cdot 0.01 \tag{46}$$

where (x_j, y_j) are all the points visited by the robot previously.

As stated in the algorithm, the choice for the reward functions $\mathcal{E}_i(.)$ is dependent on the application. It may be argued that the reward functions would have to include some kind of external payoff based on the success of the task that is not considered in the model of Algorithm 4.

The interpretation of the reward functions used in the algorithm 4 is:

- $\mathcal{E}_1(.)$ is the function for the personality trait γ_1, "courage". For action α_1 "follow the uphill gradient", the reward is $\mathcal{E}_1(s_t, \alpha_1, t) = \nabla T(x_i, y_i) \cdot \vec{u}_i$. Notice that this value is positive if the angle between the gradient ∇T and the uphill unit vector $\vec{u}_i$ is in the interval $(-90°, 90°)$ and negative otherwise. For action α_2, "follow the downhill gradient", the reward is $\mathcal{E}_1(s_t, \alpha_2, t) = \nabla T(x_i, y_i) \cdot \vec{d}_i$. Notice that this value is positive if the angle between the gradient ∇T and the downhill unit vector $\vec{d}_i$ is in the interval $(-90°, 90°)$ and negative otherwise. In other words, the personality trait "courage" returns a larger value if the direction of the movement is closer to the gradient of the target. Since $\vec{u}_i$ and $\vec{d}_i$ are constituted by a summation of the gradient of the target and the gradients of the robots (equations (39) and (40)), the direction that is closer to the danger is preferred.

Algorithm 4 Tracking of a target

1: {Initializations} Define a base and set the initial position of all robots to it. Randomly select the position of the target (X_T, Y_T) and its standard deviation σ. Set $K = \frac{100}{\sigma\sqrt{2\pi}}$ and create the gaussian field according to (37). Initialize each personality trait to a random value between $[0, 1]$. Normalize them so that $\sum_{j=1}^{3} \gamma_j = 1;\ \gamma_j \geq 0$.
2: For each robot located at the position (X_R, Y_R), define the field around it to be according to (38) with $\sigma = 4$.
3: Initialize a list $Robots \Leftarrow [1, 2, \cdots, n]$ with all robots.
4: **repeat**
5: **for** $i \in Robots$ {for all robots} **do**
6: Calculate the gradient $\nabla T(x_i, y_i)$ at the current position of the robot.
7: **for** $j \in Robots;\ j \neq i$ **do** calculate the gradient of each robot's field $\nabla R_j(x_i, y_i)$.
8: Calculate probabilities of identifying the robot in states θ_1 and θ_2 according to (41) and (42).
9: Calculate the rewards for each personality trait $\mathcal{E}_1(.)$, $\mathcal{E}_2(.)$ and $\mathcal{E}_3(.)$.
10: Calculate the expected values for the execution of each action according to (35).
11: Calculate the uphill unit vector (39) and the downhill unit vector (40). Calculate the probability of executing action α_1 (uphill) and α_2 (downhill) as described in (43). Randomly select the action to be executed using these probabilities.
12: Calculate the step for the adaptation of traits of personality according to (45).
13: Update the personality traits using the adaptation law in (44).
14: Calculate the probability of a robot being shot as in (46).
15: **if** robot i is shot **then** remove robot i it from list *Robots* and go back to base.
16: **end for**
17: **until** all robots are back to base

- $\mathcal{E}_2(.)$ is the function for the personality trait γ_2, "fear". For action α_1 "follow the uphill gradient", the reward is $\mathcal{E}_2(s_t, \alpha_1, t) = (\sum_{j \neq i} \nabla R_j(x_i, y_i)) \cdot \vec{u}_i$. Notice that this value is positive if the angle between the summation of gradients $(\sum_{j \neq i} \nabla R_j(x_i, y_i))$ and the uphill unit vector $\vec{u}_i$ is in the interval $(-90°, 90°)$ and negative otherwise. For action α_2, "follow the downhill gradient", the reward is $\mathcal{E}_2(s_t, \alpha_2, t) = (\sum_{j \neq i} \nabla R_j(x_i, y_i)) \cdot \vec{d}_i$. Notice that this value is positive if the angle between the summation of gradients $(\sum_{j \neq i} \nabla R_j(x_i, y_i))$ and the downhill unit vector $\vec{d}_i$ is in the interval $(-90°, 90°)$ and negative otherwise. In other words, the personality trait "fear" returns a larger reward for the action that moves the robot closer to other robots.
- $\mathcal{E}_3(.)$ is the function for the personality trait γ_3, "cooperation". For both actions, the reward is calculated as $\mathcal{E}_2(s_t, \alpha_k, t) = \sum_{j \neq i} \nabla R_j(x_k, y_k)$, for $k = 1, 2$ where (x_k, y_k) is the future position of the robot. Let $\vec{p}_i$ be the robots current position and $|\vec{v}_i|$ the speed of the robot, which, in our case is 1 unit/s. For action α_1 "follow the uphill gradient", the reward function $\mathcal{E}_3(.)$ is evaluated at $(x_1, y_1) = (\vec{p}_i + \vec{u}_i \cdot |\vec{v}_i|)$. For action α_2, "follow the downhill gradient", the reward function $\mathcal{E}_3(.)$ is evaluated at $(x_2, y_2) = (\vec{p}_i + \vec{d}_i \cdot |\vec{v}_i|)$. The personality trait "cooperation" assumes that when the robot moves closer to other robots, the survival of the groups is enhanced.

Indeed, the concept of success in the definition of reward functions is very subjective. Depending on the information we have available and the complexity of the model we establish, success would have a completely different meaning. For example, with the same setup of Algorithm 4, we may assume that the robots know where the target is, and the task could be just to get close to it. In this case, the reward

functions $\mathcal{E}_i(.)$ could include external information on how dangerous the environment becomes at each step, say how close a shot came to hit the robot, or how close we got to the target. Notice that in Algorithm 4, we did not assume that this information is available.

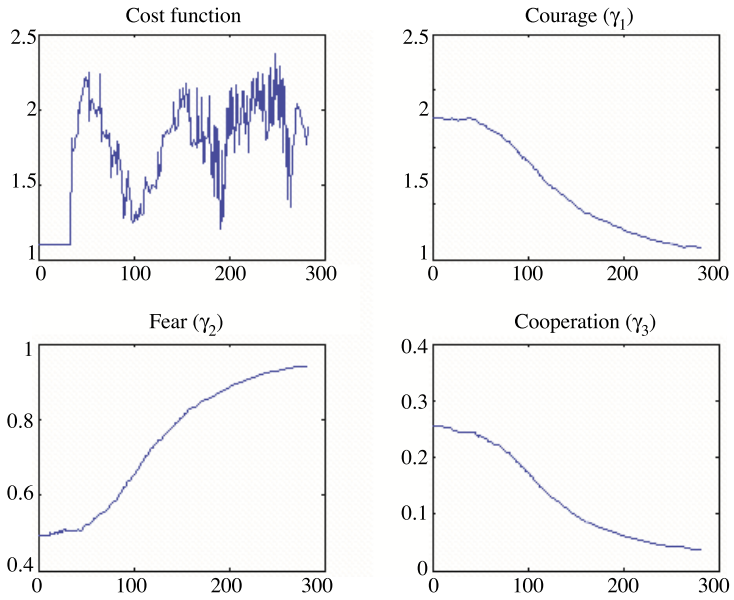

Figure 4. Utility function and personality traits of one robot

Since we chose a low value for the learning rate ($\eta = 0.01$), it is expected that there will be a slow convergence of the traits of personality to a steady state value. Results for one arbitrarily chosen robot are depicted in Figure 4. This figure indicates that the traits of personality do converge to a steady state value. In this figure, the value function is $\mathcal{U}(s_t, \bar{\gamma}, \alpha_k)$, where α_k is the action executed at time t. Notice that the value function varies around some range (this is not necessarily the case; until further proof, this should be taken as a particularity of the simulation analyzed). We may notice that the robot becomes a "fearful" robot (γ_2 increases, while the other traits decrease). Therefore, we may hypothesize that this particular robot is in some kind of cluster of robots, which makes variations on the cost functions for the particular traits more difficult. Moreover, the particular values of the personalities are characteristic of the one simulation at hand. Had we had a different initialization, we could get to different steady state values for the traits of personality, since the environment changes considerably as well as the robots initial conditions (the initial values for the traits of personality). Table 6 shows that in a given run, all the robots do converge to a steady state value and they are related to each other. This is not necessarily true for different payoff tables (like Table 5) and reward functions ($\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$) and must be considered (until further proof) as a particularity of the simulation setup under analysis.

In order to evaluate the quality of the simulations, we measure the quality of the target location by the robots. When the i^{th} robot goes back to base, it records the position (x_{S_i}, y_{S_i}) where it was shot (recall that we suppose that the robot just goes back to base when it is shot). Therefore, if (x_T, y_T) is the actual position of the target, the error of the best target location is $(\|(x_T, y_T) - (x_{S_i}, y_{S_i})\|), i = 1, \cdots, n$, where n is the number of robots in the simulation. We also measure the total time that it takes for all the robots to get back to base, and the average location error for all robots in the simulation. The results are shown in

Table 6. Convergence of the personality traits

Number of robots	Courage (γ_1)		Fear (γ_2)		Cooperation (γ_3)	
	Average	Standard deviation	Average	Standard deviation	Average	Standard deviation
10	0.1648	0.1377	0.1158	0.1435	0.7194	0.2743
20	0.2451	0.1006	0.2381	0.1713	0.5167	0.2316
30	0.1299	0.0930	0.3066	0.1827	0.5636	0.1692

Table 7. Simulation results.

Number of robots	Target location error		Total Time		Location error for all robots	
	Average	Standard deviation	Average	Standard deviation	Average	Standard deviation
10	12.2000	6.5201	93.3000	55.6698	17.3810	7.7339
20	9.5880	3.8975	136.1000	45.4715	15.5337	5.4713
50	8.4136	3.9315	322.5000	117.8740	15.3012	5.5135

Table 7, wherein we have the average and standard deviation of the target location error, total time of the missions and average location error for all robots. All the results are obtained through 10 executions of the target-tracking mission.

The results indicate that some robot behaviours are independent of the number of robots in the fleet. There is also a tendency to get a better location of the target with an increasing number of robots. This is due to our assumption that the robots are less likely to get shot when they are in larger numbers (equation (46)) by means of huddling together, which has been observed in the simulations. In fact, in order to visualize better the effect of the other robots in how a robot decides to act, we considered the enemy (the tank) to be more accurate and replaced (46) by

$$P(shot) = \frac{|\nabla T(x_i, y_i) - 10 \sum_{j \neq i} \nabla R_j(x_i, y_i)|}{\max(|T(x_j, y_j)|)} \cdot 0.1 \tag{47}$$

i.e., the robots are 10 times more likely to be shot than predicted in the algorithm (therefore the probability is multiplied by 0.1 instead of 0.01). Also, the presence of other robots in the neighbourhood makes more unlikely for a robot to be shot (this is the meaning of the factor 10 in the equation above). In this way, robots will take advantage of the increase in the number of robots in the neighbourhood. Table 7 also indicates that as the number of robots increases, the total time for target location also increases, although not linearly. This happens for two different reasons. First of all, the robots take longer to leave the base (we assume that just one robot leaves the base at each time step). Second of all, as we have a larger number of robots, the chance of being shot decreases (again, (47)) and, therefore, they take much longer to get back to base.

When we used the probability in (46) (as in the simulation illustrated in Figure 4), the traits "fear" and "cooperation" are always more important, given rise to the most interesting behaviour observed in the simulation: the tendency for the robots to huddle together. In most simulations, they formed a big group and kept like that until the individual robots were being shot by the enemy. This may also be seen in Table 7, for as we increase the number of robots, the average distance to the target slightly decreases. This is also

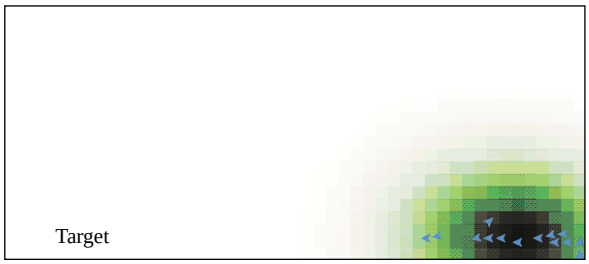

Figure 5. State of the robots during the simulation.

a result from the emergent huddling behaviour. Figure 5 shows a picture of the state of the robots in the simulation. We can see that the robots do huddle together, but some of them (the more courageous ones) move farther from the centre of the swarm. However, they are more likely to be shot (a result seen from (46)).

Another aspect observed in the simulations was the behaviour of robots after some of them were shot. Observe that, since the number of active robots decrease, the reward $\mathcal{E}_2(.)$ for the personality trait γ_2, "fear", calculated on step 19 of Algorithm 4, and $\mathcal{E}_3(.)$ for the personality trait γ_3, "cooperation", calculated on step 20, decrease. Therefore, the reward $\mathcal{E}_1(.)$ for the trait of personality γ_1, "courage", gets more important for the remaining robots and they tend to "attack" the target more directly. This was a behaviour observed when just some few robots were left. Since there is no other robot to help them, the remaining robot takes more risks and move toward the target, therefore increasing the risk of being shot.

Figure 6. State of the simulation when two robots are turned courageous.

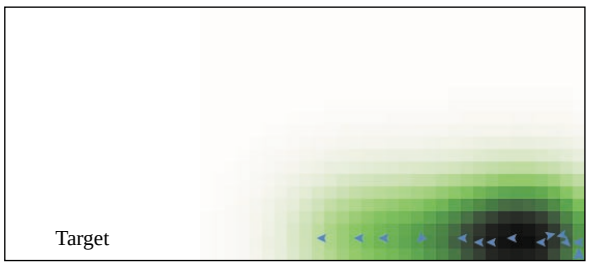

Figure 7. State of the simulation when five robots are turned courageous.

In order to examine how resilient to spurious behaviours the swarm is, we then fixed some robots

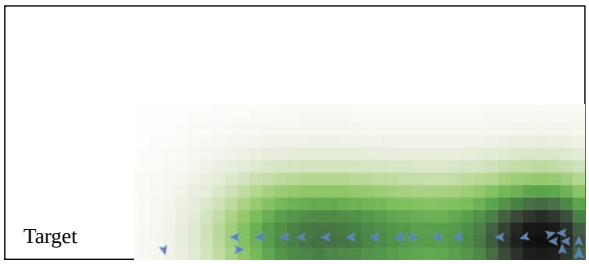

Figure 8. State of the simulation when ten robots are turned courageous.

as courageous and ran the simulation again. The idea is to check out when the group will start showing different group behaviours than the ones observed so far. Figures 6, 7 and 8 are snapshots of the simulation over the period of time when some robots were set to "courageous". We also reduced the probability of getting shot even more, to just 10% of the value in (47). Figure 6 shows the state of the simulation when 2 out of 20 robots are made courageous. It does not look very different from the state of the simulation in Figure 5. Figure 7 shows the state of the simulation when 5 out of 20 robots are made courageous and Figure 8 shows the state of the simulation when we turn 10 out of 20 robots courageous. We see that in Figure 7, the group of robots start to break and, in Figure 8, when half the robots turn courageous, the group of robots is completed broken. This suggests that the swarm of robots is resilient to outlier individuals up to some limit, but as the number of robots with some specific trait of personality increases, the swarm dynamics can dramatically change. In the case depicted in Figures 6 to 8 we see that when more robots become courageous, they drive the entire group to a courageous state. We note that for this behaviour to become noticeable we artificially and arbitrarily set the trait of personality γ_1, "courage", for the courageous robots to 1 and the other two traits were set to zero.

5. Conclusion

This paper has presented a unique method of modeling and controlling a swarm of robots. The paper integrates ideas from game theory and incorporates the novel use of adaptive personality features to achieve an intelligent swarm. Three different simulations have been presented. Each simulation scenario highlights a different aspect of swarm intelligence using game theory and adaptive personalities.

The first simulation illustrated how two agents or robots can play a zero-sum game and how the agent/robot personalities would converge to the Nash equilibrium. A proof of convergence theoretically validates the method. The second simulation is an example of three robots that must cooperate in leaving a room. It is shown how the proposed method can achieve optimal performance. Furthermore, one robot converges to the always walk condition another converges to the altruistic always wait condition and the third robot converges to the mixed 50% wait and 50% walk strategy.

The third simulation illustrates how the proposed method can be used to locate a target. The effect of different robot personalities on the performance of the swarm is shown. Cooperative robots tend to huddle into a tight swarm, whereas more courageous robots leave the swarm and lead the pack. We also demonstrate that the swarm is resilient to spurious individuals. By fixing some individuals as "aggressive", we showed that it takes up to half of the swarm to change the arising group behaviour. This is an important result, since malfunctioning robots must be dealt with.

References

[1] E. Bonabeau, M. Dorigo, G. Theraulaz. Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York, New York, 1999.

[2] M. Dorigo, V. Trianni, E. Şahin, et. al., "Evolving self-organizing behaviours for a swarm-bot". Autonomous Robots, 17: pp. 223–245, 2004.

[3] E. Şahin. "Swarm robotics: From sources of inspiration to domains of application". In E. Şahin, W. M. Spears, editors, Swarm Robotics: SAB 2004 international workshop, Santa Monica, CA, USA, July 17, 2004 : revised selected papers, pp. 10–20, Berlin, Heidelberg, 2005. Springer-Verlag.

[4] G. Beni. "From swarm intelligence to swarm robotics". In E. Şahin, W. M. Spears, editors, Swarm Robotics: SAB 2004 international workshop, Santa Monica, CA, USA, July 17, 2004 : revised selected papers, pp. 1–9, Berlin, Heidelberg, 2005. Springer-Verlag.

[5] M. Dorigo et. al., The swarm-bots project. In E. Şahin, W. M. Spears, editors, Swarm Robotics: SAB 2004 international workshop, Santa Monica, CA, USA, July 17, 2004 : revised selected papers, pp. 1–9, Berlin, Heidelberg, 2005. Springer-Verlag.

[6] M. Mynsk. The Society of Mind. Simon & Schuster, New York, New York, 1986.

[7] S. N. Givigi Jr., H. M. Schwartz. "A game theoretic approach to swarm robotics". Applied Bionics and Biomechanics, 3: pp. 1–12, 2006.

[8] D. Fudenberg, D. K. Levine. The Theory of Learning in Games. The MIT Press, Cambridge, Massachusetts, 1998.

[9] J. S. Shamma, G. Arslan. "Dynamic fictitious play, dynamic gradient play, and distributed convergence to nash equilibria". IEEE Transactions on Automatic Control, 50(3): pp. 312–327, 2005.

[10] J. W. Weibull. Evolutionary Game Theory. MIT Press, Cambridge, Massachusetts, 1995.

[11] S. Hart. "Adaptive heuristics". Econometrica, 73(5): pp. 1401–1430, 2005.

[12] G. W. Brown. "Iterative solutions of games by fictitious play". In T. C. Koopmans, editor, Activity Analysis of Production and Allocation, pp. 374–376. Wiley, New York, New York, 1951.

[13] J. Robinson. "An iterative method of solving a game". The Annals of Mathmatics, 54(2): pp. 296–301, 1951.

[14] R. S. Sutton, A. G. Barto. Reinforcement learning: an introduction. The MIT Press, Cambridge, Massachusetts, 1998.

[15] R. B. Myerson. Game Theory: Analysis of Conflict. Harvard University Press, Cambridge, Massachusetts, 1991.

[16] J. Maynard-Smith. Evolution and the Theory of Games. Cambridge University Press, Cambridge, Massachusetts, 1982.

[17] G. Weiss, editor. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press, Cambridge, Massachusetts, 1999.

[18] C. Darwin. The expression of the emotions in man and animals. University of Chicago Press, Chicago, Illinois, 1965.

[19] D. Yingying, H. Yan, J. Jing-ping. Self-organizing multi-robot system based on personality evolution. In IEEE International Conference on Systems, Man and Cybernetics, volume 5, 2002.

[20] P. D. Straffin. Game theory and strategy. The Mathematical Association of America, Washington, DC, 1993.

[21] N. N. Vorob'ev. Game Theory: Lectures for Economists and Systems Scientists. Springer-Verlag, New York, New York, 1977.

[22] J. S. Shamma, G. Arslan. "Unified convergence proofs of continuous-time fictitious play". IEEE Transactions on Automatic Control, 49(7): pp. 1137–1142, 2004.

[23] D. P. Bertsekas. Dynamic Programming and Optimal Control. Athena Scientific, Belmont, MA, 1995.

[24] L. P. Kaelbling, M. L. Littman, A. W. Moore. "Reinforcement Learning: A Survey". Journal of Artificial Intelligence Research, 4: pp.237–285, 1996.

[25] R. Dawkins. The selfish gene. Oxford University Press, New York, new ed. edition, 1989.