A game theoretic approach to swarm robotics

doi:10.1533/abbi.2005.0021

S. N. Givigi, Jr. and H. M. Schwartz

Department of Systems and Computer Engineering, Carleton University, Ottawa, Ontario, Canada

Abstract: In this article, we discuss some techniques for achieving swarm intelligent robots through the use of traits of personality. Traits of personality are characteristics of each robot that, altogether, define the robot's behaviours. We discuss the use of evolutionary psychology to select a set of traits of personality that will evolve due to a learning process based on reinforcement learning. The use of Game Theory is introduced, and some simulations showing its potential are reported.

Key words: Game theory, swarm intelligence, adaptive systems, traits of personality.

INTRODUCTION

Swarm intelligence may be defined as a distributed problem-solving technique based on self-organization theory and inspired by collective social behaviour (Bonabeau *et al.* 1999). Moreover, the main natural (or biological) base for swarm intelligence is social insect colonies, which are simple individually but present a highly complex social behaviour based on (supposedly) very simple rules (Dorigo *et al.* 2004). Also, they have almost no direct communication, but are able to overcome difficult tasks by observing changes in the environment performed by other individuals (Bonabeau *et al.* 1999; Dorigo *et al.* 2004).

In practice, swarm intelligence is not intended to generate a rational individual entity, as classical Artificial Intelligence proposes (Bonabeau *et al.* 1999). However, through investigation of simple computational models that may be implemented in simple machines, it 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 (Dorigo *et al.* 2004). 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 modelling and simulation in order to justify techniques that may be used in a swarm-based robotic environment.

Corresponding Author:
Sidney N. Givigi, Jr.
Department of Systems and Computer Engineering
1125 Colonel By Drive Carleton University
Ottawa, Ontario K1S 5B6
Tel: +1 613 520-5740; Fax: +1 613 520-5727
Email: sgivigi@sce.carleton.ca or schwartz@sce.carleton.ca

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

1. The algorithms the robots must execute are simple (computationally) and may be run by simple digital signal processor (DSP) (or even field programmable gate array (FPGA)) chips.
2. The robots' sensory capacity is limited and supposed to be the same.
3. The message exchanging is very limited or inexistent. 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.
4. The robots are able to sense the presence of other robots and, for simulation sake, the position of targets or enemies.

To observe the emergence of group behaviours, we rely on the concept of 'personality traits' (Givigi and Schwartz 2005; Mynsk 1986). 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 (Bonabeau *et al.* 1999).

SOCIAL MODELLING 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 modelling of social interactions of rational entities for the prediction of outcomes of conflicts among them (Myerson 1991). It turns out that the same approach may be used in the modelling of robot swarms, since their formation may be thought of as a social interaction of individuals (Bonabeau *et al.* 1999).

Game Theory can be defined as 'the study of mathematical models of conflict and cooperation between intelligent rational decision-makers' (Myerson 1991). 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 (Maynard 1982) and the situations they usually are involved in concern mainly conflict and cooperation.

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 analyse 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 modelling relationships between robots that are on the same side and we do not intend to model fights between groups of robots.

Our goal is to establish a theoretical framework that might lead to a stability criterion linked to the idea of equilibria in games (Weiss 1999). We also want to verify how well the inferences made inside the theory may represent learning based on the assumptions made by the framework.

REPRESENTATION OF THE ENVIRONMENT

The way the environment is represented has a great influence on which techniques may be used to control the robots as well as in how the payoff tables used by Game Theory are implemented. To reduce the robots' computational requirements, the environment is represented by potential fields in a technique known as Co-Fields (Mamei *et al.* 2004).

Co-fields are based upon cooperation through social potential fields (Chalmers *et al.* 2004). Each robot is considered to be a particle with a given position in a fixed time. The environment, containing the other robots, obstacles and enemies, is represented individually by potential fields. The potential fields represent attraction or repulsion among the objects. In the approach presented here, the meaning of the field is determined by the robot through the adaptation of personality traits as discussed in the next section. For now, we may say that the algorithm the robot is running might prefer a downhill approach (where the robot looks for a low potential valley within its representation) or an uphill approach (where the robot pursues a peak) depending on the task it has and the personalities it developed.

For each robot, according to the potential field created by any object it perceives, we may write the 'force' (Borenstein and Koren 1989) acting over it by

$$\vec{F}_T = \sum_{\text{attractive}} \vec{F}_i + \sum_{\text{repulsive}} \vec{F}_j \tag{1}$$

where, again, the attractive and repulsive forces are defined differently for each robot. As a result, each individual robot perceives the world in a different way than the others. The differences in the representations may result from the learning of new traits of personality or from noise in the sensors. Either way, the robot will have some outcome from its reading. In Equation (1), $\vec{F}_T$ is the resultant of the individual forces (attractive and repulsive) acting on the robot.

The direction of the robot's movement is the direction of the resultant force. Observe that the force may be represented as

$$\vec{F}_T = |\vec{F}_T|\vec{d}_F \tag{2}$$

where $\vec{d}_F$ is the unit vector in the direction of the resultant force. In other words, it is

$$\vec{d}_F = \frac{\vec{F}_T}{|\vec{F}_T|} \tag{3}$$

We use this notation to emphasize that the fields just define the direction of the movement, but the final decision to move in this direction and the velocity of the movement are determined by the personality traits as defined in next section.

PERSONALITY TRAITS

One of the suggestions of the Theory of Evolution is that animals have emotions (Darwin 1965). 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 (Mynsk 1986). Using these ideas, we define a way robots may react to their environment (Givigi and Schwartz 2005).

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 to better perform a complex task. But we do not want to add complexity to our algorithms. 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 that is able to specialize for executing some tasks but at the same time may learn how to execute a different action if it is necessary.

Personality traits are represented by real numbers γ_i. They are used 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

 doi:10.1533/abbi.2005.0021

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 $f : X \times Y \times \mathrm{A} \rightarrow X \times Y$, 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 an example of human beings to explain this. 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.

Since the states of the system are represented using potential fields as described in the previous section, we avoid the proliferation of actions that the robot can take. In the simulations to come, the robots will always choose the uphill or downhill algorithm (plus the choice to stay put). One thing that should be stressed here is that the uphill or downhill move does not need to be on the exact direction of the gradient, but, with variations in the personality traits, the uphill or downhill move has to keep an angle between $+90^\circ$ and -90° with the gradient. Then, when the state is recognized, the action is chosen according to the formula, known as the randomised strategy (Kaelbling 1996), which is useful for leading the robot to 'explore' new actions and not just 'exploit' learnt sequence of actions. The next equation demonstrates the randomised strategy.

$$P(\alpha_i|s) = \frac{k^{V(s,\alpha_i)/T}}{\sum_{j=1}^{n} k^{V(s,\alpha_j)/T}} \tag{4}$$

In Equation (4), P is the probability that action α_{i} will be executed when the state is s. 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 (Kaelbling 1996). The value function $V(.)$ is related to the action under consideration and the current state. $V(.)$ 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, for, since robots learn over time, they will develop the capability to 'predict' the payoffs of their actions. And finally n 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 article, 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 (Kaelbling 1996).

As stated before, this approach (Equation (4)) permits that we exploit the better solutions found so far, but also allows us to explore new possibilities. Since our states are simple and the number of actions is limited, the task to update the values $V(.)$ does not take much computational power. When we use game theory to represent the payoffs a robot receives from the environment, we need a way to represent the overall payoff a robot will get during the execution of the task. To do that, it is more cost-effective to incorporate the value function $V(.)$ in another function that considers the traits of personality. Therefore, let us define some cost functions ε_i that represent how well a personality trait contributed for the success of the robot. These functions are arbitrary and defined on the basis of the problem under consideration (Givigi and Schwartz 2005). When an action α_i is chosen to be executed, all personality traits γ are then updated.

The effect of the action taken is evaluated using the equation

$$E(\gamma_1, \gamma_2, \gamma_3, \ldots, \gamma_n, s_t, \alpha_i, t) = \sum_{j=1}^{n} \gamma_j \varepsilon_j(s_t, \alpha_i, t) \tag{5}$$

where the individual cost functions ε_j are related to how beneficial for the robot the execution of action α_{i} is according to each personality trait γ_{j} and, therefore, determines the reward and/or penalty related to the trait of personality. Furthermore, we assume that the weights γ_i (i.e. the personality traits) are normalized, so

$$\sum_{j=1}^{n} \gamma_j = 1; \quad \gamma_j \geq 0 \tag{6}$$

The action chosen is the one with the largest value

$$V(s, \alpha_i) = E(\gamma_1, \gamma_2, \ldots, s_t, \alpha_i, t) - E(\gamma_1, \gamma_2, \ldots, s_{t-1}, \alpha_j, t-1) \tag{7}$$

i.e. the one that makes the highest improvement in the overall value function. Equation (7) takes into account all the traits of personality γ_i and the environment as represented

Table 1 **Zero-sum game example**

Game 5.1.1			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

at time t (the next time step) and $t-1$ (the immediate past step). Action α_i is the action under consideration and α_j is the action taken at time step $t-1$. Finally, s_t is the state that the robot perceives the environment to be in at time t, and s_{t-1} is the state the robot perceived the environment to be in at time $t-1$. Notice that, in most applications, it is not necessary to keep the quantity $E(.)$ at time $t-1$. However, for consistency, we represent it in Equation (7).

Moreover, some procedure for evolving the personalities is necessary. Its main purpose is to diminish the dilemma between stability and adaptability (plasticity) of learning. The adaptation law used is

$$\gamma_i(t) = \gamma_i(t-1) + \eta \Delta\gamma_i(t) \tag{8}$$

where η is the learning rate for the personalities, i.e. a real value in the interval $[0, 1]$, and defining $\Delta\varepsilon_i(t) = \varepsilon_i(s_t, \alpha_j, t) - \varepsilon_i(s_{t-1}, \alpha_k, t-1)$, where α_j is the action taken at time t and α_k was the action taken at time $t-1$, the step is

$$\Delta\gamma_i(t) = \frac{\Delta\varepsilon_i(t)}{\sum_{j=1}^{n} \Delta\varepsilon_j(t)} \tag{9}$$

Equations (8) and (9) 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 Equation (8) 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 chatters 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 for a smoother convergence. The difference equation in Equation (8) is in the same form used in the reinforcement learning literature. More details may be found in (Kaelbling *et al.* 1996).

SIMULATION RESULTS

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 the section A Zero-Sum Game Example, by a zero-sum (matrix) game wherein no saddle point exists and therefore mixed strategies must be used.[1] We will compare the theoretical optimal solution with the results obtained by the learning procedure depicted in this work. In the section Robots Leaving a Room, 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 (three robots) and the reward will not be only the payoff table, but a combination of payoffs and goal achievement. Finally, in the section Tracking a Target, 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.

Table 2 **Optimal mixed strategies**

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

A zero-sum game example

The game that we will analyse in this section is reported in (Straffin 1993) and 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, it gets a reward of 4 units while player A gets penalized 4 units.

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.

[1] For a gentle introduction to game theory, refer to Straffin (1993). For a more complete reference, see Vorob'ev (1977).

 doi:10.1533/abbi.2005.0021

Table 3 **Experimental results obtained for player B**

Strategy	Optimal (%)	Experimental average (%)	Experimental standard deviation
B1	0	0	0
B2	36	18.03	36.95
B3	0	0	0
B4	57	60.49	46.60
B5	0	0	0
B6	7	21.48	40.09

We suppose that player A always plays its best strategies (24% A1, 21% A2, 55% A3) and player B starts playing all strategies equally; i.e. each strategy will have a probability of 16.67% of being used. The question we want to answer is: If we use personalities as described above, will the robot learn to play the best mixed strategy? If not, will there be any improvement over time?

The algorithm followed by player B is as follows.

Algorithm 1

1. Define the payoffs for the game to be according to the matrix

$$M = \begin{bmatrix} 4 & -4 & 3 & 2 & -3 & 3 \\ -1 & -1 & -2 & 0 & 0 & 4 \\ -1 & 2 & 1 & -1 & 2 & -2 \end{bmatrix} \quad (10)$$

2. Initialise learning rate $\eta = 0.01$.
3. Define and initialise six personality traits $\gamma_i = \frac{1}{6}$, $i = 1, \ldots, 6$.
4. Repeat steps 5 to 9 for 1,000 repeated games.
5. Player A randomly chooses one of its strategies to play using the probabilities: 24% of using strategy A1, 21% of using strategy A2, 55% of using strategy A3. The action chosen is then called a.
6. Player B randomly chooses one of its strategies, using the equation

$$P(B_i) = \frac{e^{\gamma_i}}{\sum_{j=1}^{n} e^{\gamma_j}} \quad (11)$$

   The action chosen is then called b. Equation (11) is Equation (4) with $k = e$ (exp(1) = 2.7183), $T = 1$ and $V(s, \alpha_i.) = \gamma_i$.
7. The payoff for player B is $-M_{ab}$. Therefore, update the personality trait

$$\gamma_b = \gamma_b - \eta M_{ab} \quad (12)$$

   This equation is of the same form as Equation (8); however, the step size, $-M_{ab}$, is represented directly from the payoff table.
8. Normalize all personality traits γ_i so that (implementation of Equation (6))

$$\sum_{j=1}^{6} \gamma_j = 1; \quad \gamma_j \geq 0 \quad (13)$$

9. Record the payoff for player B ($-M_{ab}$).

As seen in the algorithm above, we have defined six traits of personality, one for each strategy. 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) to characterize our player. Our goal was just to make the simulation more straightforward. Therefore, the value functions as defined in Equation (7) are $V(s, \alpha_i) = \gamma_i$. The learning rate ($\eta = 0.01$), resulted in a smooth convergence to the stable final personality traits. Moreover, we set the values in Equation (4) (Equation (11) above) to be $k = e$ (exp(1) = 2.7183) and $T = 1$, which leads to more exploitation of already-learnt strategies than exploration of new approaches (Kaelbling *et al.* 1996).

We then ran the simulation for 30 times and collected the final probabilities of using each one of the 6 strategies at player B's disposal as well as the value that the same player obtained in each simulation. The average of such results and their respective standard deviation are summarized in Table 3. One may notice that, indeed, the procedure made the personalities vary around its optimal. Furthermore, we observe that the disadvantageous strategies B1 and B3 were 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 0.0664 with a standard deviation of 0.0292, very close to the theoretical value of the game, 0.07 (Straffin 1993).

Now we ask a different question: what if player A changes its strategy, drifting away from the optimal? Will player B respond to the change? We ran another set of 30 simulation sets, but after 500 repetitions of the game player A assumes a different mixed strategy (70%, 20%, 10%), trying to take advantage of the overuse of strategies B4 and B6 by player B. In this scenario, the first 500 games presented the same results as analysed before. However, for the next 500 games, player B switches to strategy B2 almost 100% of the time in order to take advantage of player A's poor choice of strategies. As a result, the payoff of the last games is, on average, 2.63, much higher than that of the previous games. The only other strategy used was B5, but, as it is dominated by B2, it is used less than 0.001% of the time.

The importance of this example resides in the fact that the robot is able to learn from experience when the task is represented in terms of a game. Furthermore, it shows that even when the opponent changes its strategy, the learning procedure adapts the personality traits and the robot's behaviours change accordingly.

Robots leaving a room

Our second simulation is similar to the one introduced and reported in (Yingying *et al.* 2002). The set-up 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 centred at (3, 8) and with dimensions so that just one robot may pass. Inside the room, there

 doi:10.1533/abbi.2005.0021

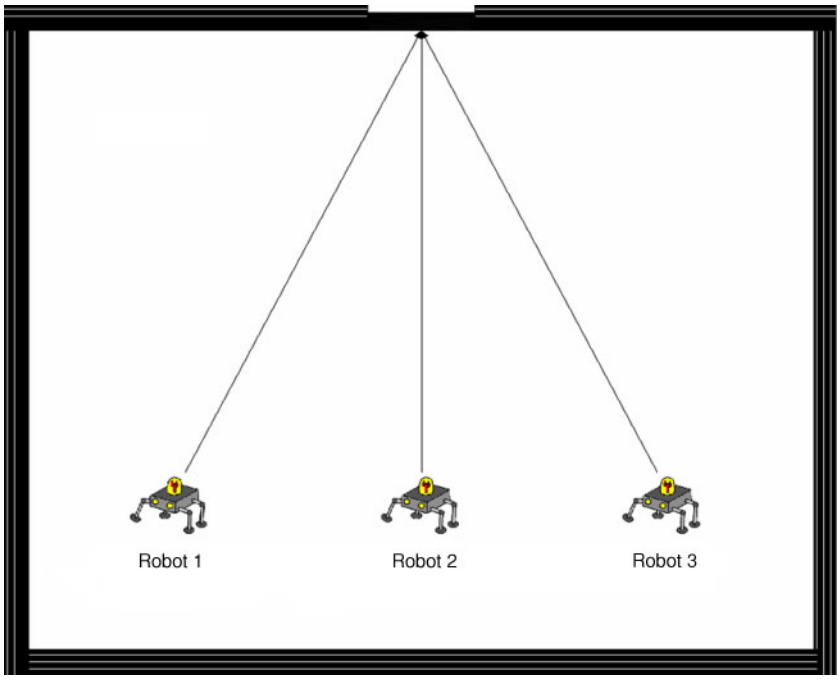

Figure 1 Artistic depiction of the problems of robots leaving a room.

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

Robots leaving a room		Player B	
		Walk	Wait
Player A	Walk	−1	X
	Wait	Y	0

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 (Fig. 1). It is assumed that one robot knows the positions of the other two without any noise, and it is also assumed that the robots move only in a straight line from their initial position toward the centre 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

$$X \in Z^{+} \qquad \text{and} \qquad Y \in Z^{-} \tag{14}$$

where the values for X and Y will depend on how important reward and penalty are in the modelling. The complete algorithm is as follows:

Algorithm 2

1. Define the payoffs for the robots 1 and 3 in Figure 1 to be according to the matrix

$$M_1 = \begin{bmatrix} -1 & 1 \\ -1 & 0 \end{bmatrix} \tag{15}$$

i.e. where $X = 1$ and $Y = -1$ (see Table 4 and Equation (14)). For player 2, define a different payoff table as

$$M_2 = \begin{bmatrix} -1 & 3 \\ -1 & 0 \end{bmatrix} \tag{16}$$

where $X = 3$ and $Y = -1$ (see Table 4 and Equation (14)). The values are different such that robot 2's expected reward is more representative than its expected penalty.

2. Initialise learning rate $\eta = 0.01$.
3. Each robot will have two personality traits, initialised as $\gamma_i = \frac{1}{2}$, $i = 1, 2$, which define which strategy (Walk or Wait in Table 4) the robot will play.
4. Repeat steps 5 to 11 for 100 repeated games. Keep a counter j with the number of the game.
5. Robots walk at the fixed speed until they get to a position they cannot move without hitting each other.
6. When they get to this position, all of them take a decision to move or to wait for the other according to their personalities. The probability to move is given in Equation (4), where, $k = e$ $(\exp(1) = 2.7183)$, $T = 1$ and $V(s, \alpha_i) = \gamma_i$. Therefore, Equation (4) becomes

$$P(\text{Walk}) = \frac{e^{\gamma_1}}{e^{\gamma_1} + e^{\gamma_2}} \tag{17}$$

The probability the robot will wait for the other is

$$P(\text{Wait}) = 1 - P(\text{Walk}) \tag{18}$$

The action taken by each robot is represented by A_l, where $l = 1, 2, 3$.

7. For each robot $l = 1, 2, 3$, execute step 8.
8. If all the robots are still in the room, each one of them will play two games. If the robot has already left the

 doi:10.1533/abbi.2005.0021

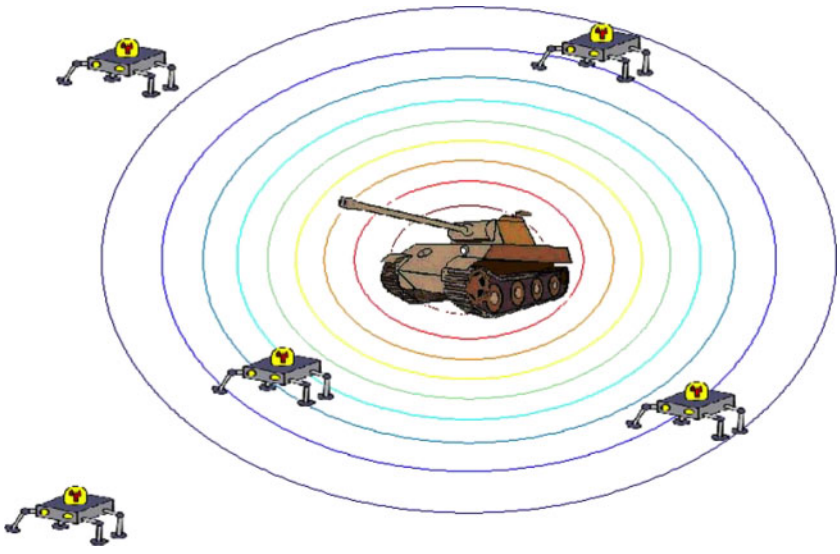

Figure 2 Artistic depiction of the simulation environment.

room, it will not play. If $l = 1$ or $l = 3$ (the robots 1 and 3 in Fig. 1), the robot gets its payoffs according to the matrix shown in Equation (15). If $l = 2$ (robot 2 in the middle in Fig. 1), the robot gets the rewards according to Equation (16). The payoffs will be represented by R_{lm}, where l is the index of the robot and m is the index of the opponent robot. For example, R_{12} is the payoff for robot 1 against robot 2. Notice that $R_{12} \neq R_{21}$.

9. If there is no collision, the robot moves; otherwise, it keeps its current position for one time step.
10. For every game played, update the personality related to the action chosen according to the equation

$$\gamma_{A_l} = \gamma_{A_l} + \eta R_{lm} \tag{19}$$

Equation (19) is the implementation of Equation (8). Notice that the personality trait is updated two times per time step.

11. Normalize all personality traits γ_i so that (implementation of Equation (6))

$$\sum_{k=1}^{2} \gamma_k = 1; \quad \gamma_k \geq 0 \tag{20}$$

The results obtained after these 100 repetitions were very interesting. 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). Secondly, the robot in the middle chooses its actions on a 50–50 basis. As a result, the average of the 100 games is 10.04s and standard deviation of 1.82. Also, 24 out of the 100 repetitions obtained the best solution of 8s (Yingying *et al.* 2002).

One may notice that the robots did not work only on 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.

In another version of the simulation shown, we considered another payoff for the robots that included how well they had performed the task of leaving the room. Knowing that the best solution is 8s, we defined payoffs of the form $\frac{1}{t-8+1}$, where t is the time the robots took to leave the room, that were added to the list of actions taken by each robot. However, this did not result in a noticeable improvement of convergence and was dropped. The result suggests that the approach proposed works well with only local information and global information (total time to leave the room) did not increase the overall convergence of the algorithm.

Tracking a target

We now make use of all the ideas presented in this article, 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 2. In this figure, we depict a target (a tank) and several robots that 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 base's 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 in comparison with 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

$$P(\theta_1|s) = \frac{|F_T|}{|F_{\text{Max}}|} - \sigma(\gamma_1, \gamma_2, \gamma_3) \tag{21}$$

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

$$\sigma(\gamma_1, \gamma_2, \gamma_3) = \gamma_1 - \gamma_2 - \gamma_3 \tag{22}$$

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 themα_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 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, and when the robot finds itself in the 'low risk' state θ_2, the robot would prefer to execute action α_1. Later on (in Equation (23)), 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.

Table 5 **Utility payoffs for states**

Utility payoffs	States	
	θ_1	θ_2
Actions		
α_1	−1	5
α_2	4	−2

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. Notice that the set of other robots may be incomplete.

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.) (Dawkins 1989).

To choose an action, we use the value function $V: X \times A \rightarrow R$ in Equation (23), which 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 $E\{V|(s, \alpha_i)\}$ for all possible actions α_i. We can think of this as a game against nature (Straffin 1993), in which the environment is supposed to play with a mixed strategy $P(\theta_i|s)$. Therefore,

 doi:10.1533/abbi.2005.0021

the expected outcome of the game in Table 5 is

$$\begin{aligned} V(s,\alpha_1) = E\{V|(s,\alpha_1)\} &= -1(P(\theta_1|s)E(\bar{\gamma},s,\alpha_1,t)) \\ &\quad + 5(P(\theta_2|s)E(\bar{\gamma},s,\alpha_1,t)) \\ V(s,\alpha_2) = E\{V|(s,\alpha_2)\} &= 4(P(\theta_1|s)E(\bar{\gamma},s,\alpha_2,t)) \\ &\quad - 2(P(\theta_2|s)E(\bar{\gamma},s,\alpha_2,t)) \end{aligned} \tag{23}$$

Equation (23) is the application of game theory expectation calculation to the framework introduced in the section Personality Traits. Actually, this equation is just the implementation of Equation (7) in terms of game theory.

Then, the action is chosen randomly on the basis of the probability (Equation (4), where $k = e(\exp(1) = 2.7183)$, $T = 1$)

$$P(\alpha_i) = \frac{e^{V(s,\alpha_i)}}{\sum_{j=1}^{2} e^{V(s,\alpha_j)}} \tag{24}$$

The complete algorithm for the simulation is as follows.

Algorithm 3

1. Initialise a rectangular space of dimensions 47 × 41 units.
2. Define the base to be in the left lower corner of the rectangular space. Set the initial position of all robots to the base.
3. Randomly select the position of the target (X_T, Y_T) anywhere inside the rectangular space as well as the standard deviation σ (this value must be chosen between the values 10 and 20), define the Gaussian field
$$T(x,y) = Ke^{-\frac{1}{2}\frac{(x-X_T)^2}{\sigma^2}} e^{-\frac{1}{2}\frac{(y-Y_T)^2}{\sigma^2}} \tag{25}$$
where K is a term to scale the sensitivity of the robots. We chose it to be $K = \frac{100}{\sigma\sqrt{2\pi}}$.
4. Initialise each personality trait to a random value between [0, 1]. Normalize them so that they add up to 1, i.e. $\sum_{j=1}^{3} \gamma_j = 1; \gamma_j \geq 0$. Set the learning rate $\eta = 0.01$.
5. When the robot is located at the position (X_R, Y_R), define the field around each robot to be
$$R(x,y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(x-X_R)^2}{\sigma^2}} e^{-\frac{1}{2}\frac{(y-Y_R)^2}{\sigma^2}} \tag{26}$$
where $\sigma = 4$ for every robot.
6. For each time step t, repeat all the steps below.
7. For each robot $i = 1, \ldots, n$, repeat steps 8 to 22.
8. Get the position (x_i, y_i) of robot i.
9. Calculate the gradient $\nabla T(x_i, y_i) = \frac{\partial T(x_i,y_i)}{\partial x}\vec{i} + \frac{\partial T(x_i,y_i)}{\partial y}\vec{j}$ at the current position of the robot.
10. Calculate the gradient of each robot's field $\nabla R_j = \frac{\partial R_j(x_i,y_i)}{\partial x}\vec{i} + \frac{\partial R_j(x_i,y_i)}{\partial y}\vec{j}$ at the current position of the robot
11. Each robot has at its disposal two possible actions (or strategies). Strategy α_1 follows the direction of the uphill unit vector
$$\vec{u}_i = \frac{\nabla T(x_i,y_i) + \sum_{j\neq i} \nabla R_j(x_i,y_i)}{\left|\nabla T(x_i,y_i) + \sum_{j\neq i} \nabla R_j(x_i,y_i)\right|}$$
Strategy α_2 follows the direction of the downhill unit vector
$$\vec{d}_i = -\frac{\nabla T(x_i,y_i) + \sum_{j\neq i} \nabla R_j(x_i,y_i)}{\left|\nabla T(x_i,y_i) + \sum_{j\neq i} \nabla R_j(x_i,y_i)\right|}$$
12. Calculate
$$P(\theta_1|s_t) = \frac{|T(x_i,y_i) - \sum_{j\neq i} R_j(x_i,y_i)|}{|F_{\max}|} - \gamma_1 + \gamma_2 + \gamma_3$$
where $|F_{\max}|$ is the maximum field found so far. Set $P(\theta_2|s_t) = 1 - P(\theta_1|s_t)$.
13. Calculate the reward $\varepsilon_1(\cdot)$ for the personality trait γ_1, 'courage'. For action α_1 'follow the uphill gradient', the reward is $\varepsilon_1(s_t, \alpha_1, t) = \nabla T(x_i, y_i) \cdot \vec{u}_i$. 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 $\varepsilon_1(s_t, \alpha_2, t) = \nabla T(x_i, y_i) \cdot \vec{d}_i$. 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 (step 11 of the algorithm), the direction that is closer to the danger is preferred.
14. Calculate the reward $\varepsilon_2(.)$ for the personality trait γ_2, 'fear'. For action α_1 'follow the uphill gradient', the reward is $\varepsilon_2(s_t, \alpha_1, t) = \left(\sum_{j\neq i} \nabla R_j(x_i, y_i)\right) \cdot \vec{u}_i$. Notice that this value is positive if the angle between the summation of gradients $\left(\sum_{j\neq i} \nabla R_j(x_i, y_i)\right)$ 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 $\varepsilon_2(s_t, \alpha_2, t) = \left(\sum_{j\neq i} \nabla R_j(x_i, y_i)\right) \cdot \vec{d}_i$. Notice that this value is positive if the angle between the summation of gradients $\left(\sum_{j\neq i} \nabla R_j(x_i, y_i)\right)$ 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.
15. Calculate the reward $\varepsilon_3(.)$ for the personality trait γ_3, 'cooperation'. For both actions, the reward is calculated as $\varepsilon_3(s_t, \alpha_k, t) = \sum_{j\neq i} R_j(x_k, y_k)$, for $k = 1, 2$, where (x_k, y_k) is the future position of the robot. Let $\vec{p}_i$ be the robot's 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 $\varepsilon_3(.)$ 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 $\varepsilon_3(.)$ 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.

 doi:10.1533/abbi.2005.0021

16. Calculate the overall cost functions

$$E_1 = E(\bar{\gamma}, s_t, \alpha_1, t) = \sum_{j=1}^{3} \gamma_j \varepsilon_j(s_t, \alpha_1, t) \tag{27}$$

and

$$E_2 = E(\bar{\gamma}, s_t, \alpha_2, t) = \sum_{j=1}^{3} \gamma_j \varepsilon_j(s_t, \alpha_2, t) \tag{28}$$

Equations (30) and (31) are implementations of Equation (5).

17. Calculate the expected values for the execution of each action

$$\begin{aligned} V(s, \alpha_1) &= E\{V|(s, \alpha_1)\} = (-P(\theta_1|s) + 5P(\theta_2|s))E_1 \\ V(s, \alpha_2) &= E\{V|(s, \alpha_2)\} = (4P(\theta_1|s) - 2P(\theta_2|s))E_2 \end{aligned} \tag{29}$$

Equation (32) is the expectation of a game against nature (Straffin 1993). Actually, this is exactly Equation (23), which is Equation (7). The difference in Equation (7) is not necessary in here, since the greatest value $V(s, \alpha_i)$ will always maximize the difference predicted in Equation (7). This would not be true if other value functions (not based on game theory) were considered.

18. Calculate the probability of executing action α_1 as

$$P(\alpha_1) = \frac{e^{V(s,\alpha_1)}}{e^{V(s,\alpha_1)} + e^{V(s,\alpha_2)}} \tag{30}$$

Set $P(\alpha_2) = 1 - P(\alpha_1)$. Equation (33) is Equation (4), where $k = e$, $T = 1$ and $n = 2$.

19. Randomly select the function to be executed using the probabilities in step 18.

20. Calculate the step for the adaptation of traits of personality (Equation (9))

$$\Delta\gamma_i(t) = \frac{\Delta\varepsilon_i(t)}{\sum_{j=1}^{n} \Delta\varepsilon_j(t)} \tag{31}$$

21. Update the personality traits using the adaptation law (Equation (8))

$$\gamma_i(t) = \gamma_i(t-1) + \eta\Delta\gamma_i(t) \tag{32}$$

22. Calculate the probability of a robot being shot as

$$P(\text{shot}) = \frac{T(x_i, y_i) - \sum_{j \neq i} R_j(x_i, y_i)}{\max(T(x_i, y_i))} \cdot 0.01 \tag{33}$$

If the robot is shot, go back to base.

23. When all robots are back to base, end the execution.

Since we chose a low value for the learning rate ($\eta = 0.01$), it is expected to have a slow convergence of the traits of personality to a steady state value. Results for one arbitrarily chosen robot are depicted in Figure 3. This figure indicates that the traits of personality do converge to a steady state value. In this figure, the value function is $V(s_t, \alpha_k)$, 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 analysed). 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 initialisation, we could have got 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 (ε_1, ε_2, ε_3) and must be considered (until further proof) as a particularity of the simulation set-up under analysis.

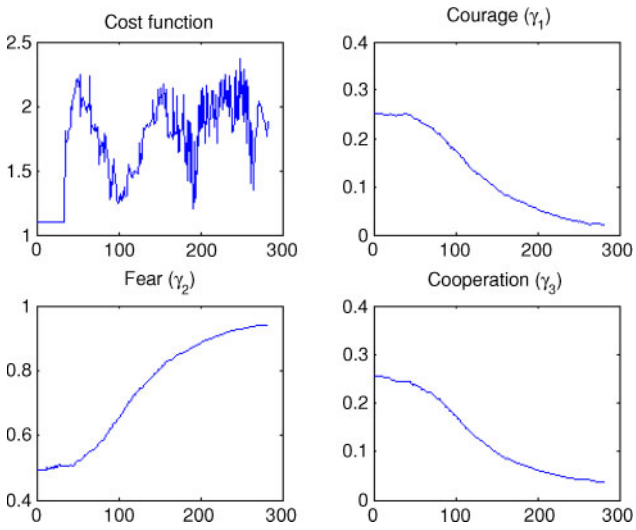

Figure 3 Value function and personality traits of one robot.

To evaluate the quality of the simulations, we measure the quality of the target location by the robots. When the ith 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 $\min(\|(x_T, y_T) - (x_{S_i}, y_{S_i})\|)$, $i = 1, \ldots, n$, 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 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 (36)) by means of huddling

 doi:10.1533/abbi.2005.0021

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

together, which has been observed in the simulations. In fact, 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 Equation (36) by

$$P(\text{shot}) = \frac{T(x_i, y_i) - 10 \sum_{j \neq i} R_j(x_i, y_i)}{\max(T(x_i, y_i))} \cdot 0.1 \quad (34)$$

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 it 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). Secondly, as we have a larger number of robots, the chance of being shot decreases (again, Equation (37)) and, therefore, they take much longer to get back to base.

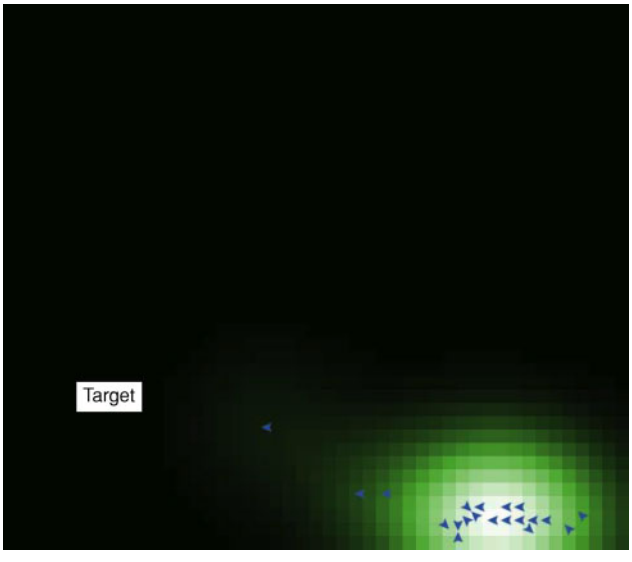

Figure 4 State of the robots during the simulation.

Another aspect observed in the simulations was the behaviour of robots after some of them were shot. Since the reward $\varepsilon_2(.)$ for the personality trait γ_2, 'fear', calculated on step 14 of Algorithm 3, decreases (since the number of active robots decreases), the reward $\varepsilon_1(.)$ for the trait of personality γ_1, 'courage', gets more important for the remaining robots and they 'attack' the target more directly. However, when we used the probability on Equation (36) (as in the simulation illustrated in Fig. 3), 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 a result from the emergent huddling behaviour. Figure 4 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 Equation (36)).

CONCLUSION

Biologically, we observe that animals react to stimuli they receive from their environment. However, the way the reaction takes place is still very debatable. Some suggest that what causes it is learning (especially when humans

are involved, which gives rise to the definition of *culture*), while others argue that hereditary (genetic) traits are more important. Without taking sides, we suggest that a mix of both is in action when animal behaviours are taken into account. Therefore, we introduce in this article the idea of learning behaviours based on parameters, called personality traits (Yingying *et al.* 2002), that are predefined.

To do that, we introduce the combination of personality traits with game theory (Givigi and Schwartz 2005). We also link these ideas with the representation of the environment using potential fields (Mamei *et al.* 2004). Moreover, methods borrowed from reinforcement learning (Kaelbling *et al.* 1996) and swarm intelligence (Bonabeau 1999) are used to create a general algorithm for traits' adaptation. We complement the meta-algorithm with its realization in several particular simulations.

The simulations shown indicate that the modelling with game theory, together with the use of personality traits that determine how robots will behave, is a powerful tool to solve a wide variety of problems. We saw that the problems ranging from zero-sum games to more complex simulations were successfully solved. The first simulation, the zero-sum game, shows that the technique may be successfully applied to games. Moreover, the outcome of the game tends to the theoretical optimal. The second simulation, the robots leaving a room, shows that both competition and cooperation may be modelled in the form of a game (Straffin 1993). Finally, the third simulation, robots tracking a target, shows how conflicting influences may lead to the adoption of compromising strategies.

Furthermore, the emergence of individual behaviours (the cooperative and competitive robots in the simulation of robots leaving a room; and the tendency for the robots to perform the task by huddling together and tracking the target in a group in the last simulation) demonstrates that even a small number of personality traits may result in quite complex group behaviours.

REFERENCES

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

Borenstein J, Koren Y. 1989. Real time obstacle avoidance for fast mobile robots. *IEEE Trans. Syst Man Cybernetics*, 19(5): 1179–1187.

Chalmers R, Scheidt D, Neighoff T, *et al.* 2004. Cooperating unmanned vehicles. In AIAA 1st Intelligent Systems Technical Conference, AIAA, pp. 2004–6252.

Darwin C. 1965. The Expression of the Emotions in Man and Animals. Chicago: University of Chicago Press.

Dawkins R. 1989. The Selfish Gene, new ed. Oxford and New York: Oxford University Press.

Dorigo M, Trianni V, Sahin E, *et al.* 2004. Evolving self-organizing behaviours for a swarm-bot. *Autonomous Robots,* 17:223–245.

Givigi SN Jr., Schwartz HM. 2005. Evolutionary swarm intelligence applied to robotics. In Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA2005), Niagara Falls, Ontario, Canada, July 29 – August 1, 2005, pp. 1005–1010.

Kaelbling LP, Littman ML, Moore AW. 1996. Reinforcement learning: A survey. *J Artif Intell Res*, 4:237–285.

Mamei M, Zambonelli F, Leonardi L. 2004. Cofields: A physically inspired approach to motion coordination. *IEEE Pervasive Comput*, 3(2):52–61.

Myerson R. 1991. Game Theory: Analysis of Conflict. London, England: Harvard University Press.

Mynsk M. 1986. The Society of Mind. New York: Simon & Schuster.

Smith MJ. 1982. Evolution and the Theory of Games. Oxford, Great Britain: Cambridge University Press.

Straffin P. 1993. Game Theory and Strategy. Washington, DC: The Mathematical Association of America.

Vorob'ev NN. 1977. Game Theory: Lectured for Economists and Systems Scientists. New York: Springer-Verlag.

Weiss G, ed. 1999. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Cambridge, MA: The MIT Press.

Yingying D, Yan H, Jing-ping J. 2002. Self-organizing multi-robot system based on personality evolution. IEEE Int Conf Syst Man Cybernetics, 5:4.

 doi:10.1533/abbi.2005.0021