Dynamic Scripting with Team Coordination in Air Combat Simulation

Armon Toubman[1,2], Jan Joris Roessingh[1], Pieter Spronck[2], Aske Plaat[2], and Jaap van den Herik[2]

[1]National Aerospace Laboratory,
Department of Training, Simulation, and Operator Performance
Anthony Fokkerweg 2, 1059 CM Amsterdam, the Netherlands
{Armon.Toubman, Jan.Joris.Roessingh}@nlr.nl
[2]Tilburg center for Cognition and Communication (TiCC), Tilburg University,
P.O. Box 90153, 5000 LE Tilburg, the Netherlands
{p.spronck, aske.plaat, jaapvandenherik}@gmail.com

Abstract. Traditionally, behavior of Computer Generated Forces (CGFs) is controlled through scripts. Building such scripts requires time and expertise, and becomes harder as the domain becomes richer and more life-like. These downsides can be reduced by automatically generating behavior for CGFs using machine learning techniques. This paper focuses on Dynamic Scripting (DS), a technique tailored to generating agent behavior. DS searches for an optimal combination of rules from a rule base. Under the assumption that intra-team coordination leads to more effective learning, we propose an extension of DS, called DS+C, with explicit coordination. In a comparison with regular DS we find that the addition of team coordination results in earlier convergence to optimal behavior. In addition, we achieved a performance increase of 20% against an unpredictable opponent. With DS+C, behavior for CGFs can be generated that is more effective since the CGFs act on knowledge achieved by coordination and the behavior converges more efficiently than with regular DS.

Keywords: computer generated forces, machine learning, air combat

1 Introduction

Military organizations are increasingly using simulations for training purposes. Simulations are cheaper, safer, and more flexible than training with real equipment in real-life situations [1, 2]. In military simulations, the roles of allies and adversaries are performed by computer generated forces (CGFs).

Traditionally, the behavior of CGFs is scripted [3]. Production rules—rules that map conditions to actions—are manually crafted to suit specific (types of) CGFs. In complex domains, such as that of air combat, this leads to complex scripts and requires availability of domain expertise. These scripts then produce rigid behavior, because it is impossible to account for all situations that CGFs might encounter during simulations.

Artificial Intelligence techniques may provide a solution by automating the process of generating CGF behavior. Automation bypasses the requirement of expertise availability and shortens the time needed to generate the behavior. Various efforts have been made at realizing automatic generation of CGF behavior [4, 5].

At the National Aerospace Laboratory (NLR) in the Netherlands, CGF research aims to generate behavior for air combat training simulations. The focus of recent work has been to generate behavior through the use of cognitive models, and optimizing these models with machine learning (ML) techniques such as neural networks and evolutionary learning [3], [6]. In this paper, we diverge from the earlier approach of using cognitive models by applying ML directly to the generation of behavior.

The envisaged new ML technique should satisfy at least four conditions to be suitable for our domain. First, the technique should provide *transparent behavior models* as a result. Techniques such as neural networks are opaque in the sense that the resulting models are hard to relate to the behavior they produce. The new technique should produce understandable models that are manually editable and reusable by training instructors. Second, the new technique should be *scalable* to the domain of air combat with team missions. The scope of the mentioned research with cognitive models [3], [6] was not scalable, since it was limited to a single learning agent. Third, the chosen machine learning technique should *converge to practically usable behavior* in a timely fashion, to allow rapid development of new training scenarios. Fourth, the ML technique must be able *to learn robust behavior*. Since the CGFs will be used for training humans, the CGFs should have good performance against a variety of tactics.

Dynamic Scripting (DS) is a reinforcement learning technique specifically designed to satisfy requirements similar as the ones stated above [7]. While DS has been used with teams, no attention has been given to the explicit coordination of teams using DS. In this paper, we present a technique based on DS called DS+C, which enables team coordination using DS through direct communication between agents. We compare the performance of a team using DS with and without coordination. The main contributions of this paper are that we (1) present explicit coordination in DS and (2) show, using an existing combat simulator, experimental evidence that coordination leads to faster convergence to optimal behavior.

The course of the paper is as follows. Section 2 describes the Dynamic Scripting method. Section 3 describes our method of team coordination. Section 4 describes a case study. Section 5 shows the results. Finally, the paper is concluded by a discussion in Section 6 and a conclusion in Section 7.

2 Dynamic Scripting Method and Related Work

Dynamic Scripting is an online learning technique based on reinforcement learning. It was introduced by Spronck et al. [7] to address certain requirements for adaptive game AI in commercial video games, such as “easily interpretable results” and “reasonably successful behavior at all times.” These requirements are also applicable in the domain of military training, where quality controls such as transparent results and robust behavior are important.

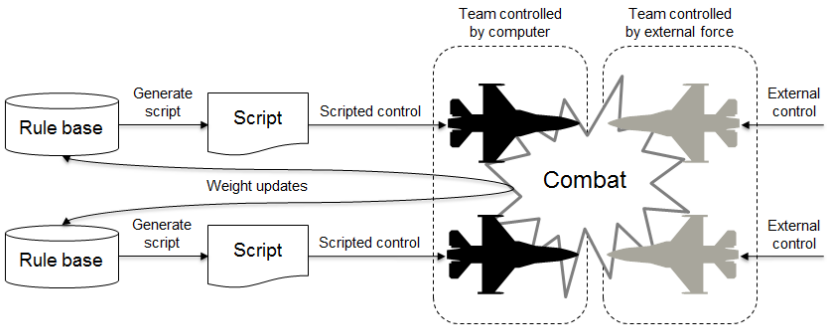

Fig. 1. Dynamic Scripting in the context of two learning agents in the air combat domain.

In DS, the learning process works as follows. The learning agent has a rule base with behavior rules. The DS algorithm selects a set of rules through weighted random selection. The selected rules together form a script that governs the behavior of the agent during a trial with one or more other agents. After each trial, the weights of the rules that were activated in the encounter are updated. The learning process is illustrated in Fig. 1.

In the original DS experiments [7] team behavior was a result of emergence, guided by a fitness function which rewarded team victories as well as individual success. However, to make sure that CGFs act conforming to the training goals of a particular air combat training simulation, more control over the team members' actions is required. Such intensive control can be formalized by coordination rules.

There are two general methods of team coordination: centralized and decentralized coordination [8]. With centralized coordination, one agent may direct the actions of a team. With decentralized coordination, all agents in a team may influence each other's actions by sharing information through some form of communication.

In this paper, we have chosen to implement decentralized control, because of its straightforward implementation. In terms of DS, decentralized control translates to each agent having their own rule base with its own weights and generated scripts. Coordination is achieved through communication. However, adding communication to multi-agent systems in general is not trivial [9]. For this reason, we attempted to fit the communication (and therefore also the coordination) explicitly into the DS mechanism.

3 Dynamic Scripting with Team Coordination

We implemented team coordination in DS through communication between agents, resulting in a technique which we called DS+C. In brief, the technique works as follows. Each agent starts with certain rules in their rule bases that are activated when particular messages are received. Whenever an agent activates a rule, it sends a message to its teammates describing its actions. The description of actions should not be

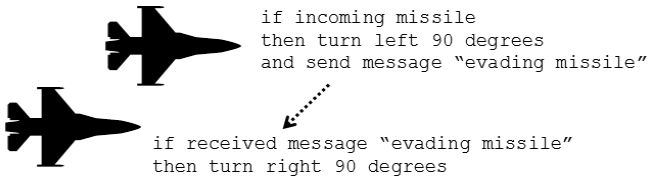

Fig. 2. Illustration of the communication applied in DS+C. Messages sent by one agent trigger rules in another agent.

too narrow; otherwise no match will occur during trials. The DS+C algorithm decides which actions in response to the messages are valuable.

In more detail, the communication scheme consists of three parts. The first part is an addition to existing behavior rules: each rule, when activated by an agent *a*, now also sends a message from agent *a* to every agent *b* in the same team. This message contains the nature of the actions described by the rule. The second part is a new component for the agents. Agent *b* stores the messages received during the activation of rules by its teammates until *b* has processed its own rules. The third part handles the processing of the received messages. For each agent, rules (i.e., the 'coordination rules') are added to its rule base that will lead to new behavior after aforementioned messages have been received. Together, these parts form a robust communication system that will remain functioning even when the recombination of behavior rules detects conflicting messages. The communication principle is shown in Fig. 2.

It must be emphasized that the form of coordination as described above is completely rule-based. The coordination rules undergo the same selection process as all other behavior rules. In other words, by expressing the coordination as rules, DS+C will learn which messages are relevant and how they should be acted upon. The rule selection part of the DS+C algorithm will include or exclude a subset of these coordination rules in scripts based on their added value.

4 Case Study and Experimental Setup

In order to test the suitability of the approach, it has been applied in the domain of air combat. In this domain, agents must exhibit realistic tactical behavior in order to increase the value of simulation training for fighter pilots.

We have taken a 'two versus one' combat engagement scenario as our testing ground. The scenario is illustrated in Fig. 3. Two 'blue' fighters (virtual pilots controlling fighter planes), i.e., a 'lead' together with its 'wingman' attempt to penetrate the enemy airspace. In more detail, the 'blue' formation seeks an engagement with a 'red' fighter that defends a volume of airspace, by

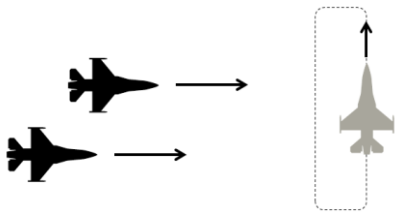

Fig. 3. Diagram of the scenario used in the case study. The 'blues' (left) fly towards the 'red' (right). Red is flying a CAP.

flying a so-called Combat Air Patrol (CAP) pattern. The 'blue' mission is considered successful (a win) if 'red' is eliminated, and is considered unsuccessful (a loss) if one or both of the 'blue' aircraft are eliminated, after which the 'blue' mission will be aborted. 'Rules of Engagement' for the 'red' fighter dictate that it will intercept fighter aircraft that fly in its direction.

The behavior of the 'blue' agents is governed by scripts generated by the newly implemented DS+C. The rule bases of the 'blue' agents contain three sets of rules. The first set consists of default rules. The default rules define basic behavior, on which the agents can fall back if no other rules apply. The rules are included in every script, and their weights will not be changed by the DS+C process. The rules also define the 'missions' of the agents; for instance, the 'blues' have default rules that let them fly to 'red' in formation, while 'red' has default rules that let it fly its CAP. The second set consists of general rules for air combat. These rules are based on domain knowledge, although highly simplified to illustrate the principles. Two instances are 'if I see an enemy on my radar, I lock this enemy with my radar' and 'if the enemy is locked by my radar, I fire a missile'. The third set consists of coordination rules. In the case of DS+C, these are the rules that produce behavior in response to the reception of certain messages. However, in the case of regular DS, these rules are 'filler' rules; rules that cannot be activated and therefore produce no behavior. These 'filler' rules were added to keep the sizes of the rule bases constant between the DS and DS+C, thus providing a fair comparison. The scripts generated by DS+C consist of 6 rules, to which the default rules were added. All rules started with a weight of 50. In total, the rule bases had 31 rules each.[1]

The 'red' agent used three basic tactics, implemented as three static scripts. The three tactics are called *Default*, a basic CAP where 'red' fires on enemies it detects; *Evading*, like *Default* but with evasive maneuvers; and *Close Range*, like *Default* but only firing from close range. These three tactics each had alternative versions in which 'red' would start the engagement from flying the CAP in the clockwise direction, rather than the counter-clockwise direction. To test whether the 'blues' would be able to learn generalized behavior, 'red' was given a composite tactic that consists of the three basic tactics plus their alternative versions. With this seventh tactic (henceforth called mixed tactics) 'red' randomly selects one of the six basic tactics and uses that tactic until it loses, at which point it would select a new tactic at random.

The performance of the 'blues' in a trial is measured using the following fitness function:

$$fitness = (0.25 + (0.5 * winner)) + 0.125 * speed + 0.125 * resources \quad (1)$$

In Eq. 1, *winner* is 1 if the 'blues' won, while it is 0 if they lost; *speed* is 1 minus the ratio of the maximum duration of a trial and the actual duration to complete the trial; and *resources* is a value between 0 and 1 based on the number of missiles spent in the trial (the idea is to learn to defeat the opponent using the least number of missiles). The fitness function is used to calculate the adjustments to the weights as follows:

[1] Descriptions of the rules are omitted for brevity, but will be published in a technical report.

$$adjustment = \max(50 * ((fitness * 2.0) - 1.0), -25) \quad (2)$$

The constants in these equations represent the balance between reward and punishment; for example, the constant -25 in equation (2) is the maximum negative adjustment after a loss, such that the associated rules with an initial weight of 50 still have some selection probability in a subsequent trial.

With DS+C, agents have additional rules in their rule bases, which lead to a larger number of possible scripts. In practice, this would lead to more trials needed to converge to successful behavior. However, since additional rules provide more options to the agents, there are also more possibilities to find optimal behavior, even in a rapid way. Below, we compare the performance of DS+C to that of regular DS. To do so, we first define performance in terms of efficiency (learning speed) and effectiveness (combat results). We define effectiveness as the mean win/loss ratio during a learning episode. It is difficult to define the efficiency of the DS algorithm, because it is hard to establish precisely when stationary performance, i.e., no further improvement takes place, is reached during learning. Both the DS algorithm and the agent environment are stochastic by nature. Therefore it is unlikely that DS converges to a single winning script. It is more likely that there is a set of sufficiently successful scripts available for a variety of situations.

To cope with the inherent variations in the learning process, we define the Turning Point (TP) measure TP(X) (based on Spronck's TP measure [7]) as the trial after which the 'blues' have won X percent of the last 20 trials. The window size (20 trials) was chosen to allow for a sufficient number of evaluation points during a learning episode (in this case 250 trials). X thus represents the chance that a winning script will be selected at that point. An early TP now represents a more efficient learning process, while a late TP represents a less efficient learning process.

Two series of experiments were run. In the first series, the 'blues' used the regular DS algorithm. In the second series, the 'blues' used DS+C. 'Red' used one of the seven tactics described in this section. The results of the experiments are described in the next section.

5 Results

For each basic tactic used by 'red', results were averaged over ten learning episodes, a learning episode representing the learning process of the 'blue' agents from zero to 250 trials (encounters). In the case of the mixed tactics, results were averaged over one hundred learning episodes to reduce noise (and thus improving the chances to observe a difference between DS and DS+C agents).

The average TPs at different percentages (50%, 60%, 70% and 80%) were calculated against each of the tactics of 'red'. For the mixed tactics, the TPs were compared using independent two-sample t-tests. Learning curves (Fig. 4) were created using a rolling average (with a window size of 20 trials) of the win/loss ratio. Additionally, the weights of all rules were recorded to check to what extent coordination rules were selected by the DS+C agents.

Table 1. TPs of DS and DS+C against mixed tactics (averaged over one hundred episodes) and the basic tactics (aggregated results, ten episodes per tactic).

		TP(50%)		TP(60%)		TP(70%)		TP(80%)	
Tactics of 'red'	**DS**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**	**μ**	**σ**
Mixed	DS	83.8	78.1	94.5	78.9	110.5	78.4	129.9	79.1
Mixed	DS+C	48.4	48.4	60.9	49.6	75.8	55.5	103.9	69.7
Basic (aggregated)	DS	55.8	56.7	66.1	57.8	87.3	66.5	122.4	82
Basic (aggregated)	DS+C	34.8	31.3	48	42.7	65	61.3	90.7	80.6

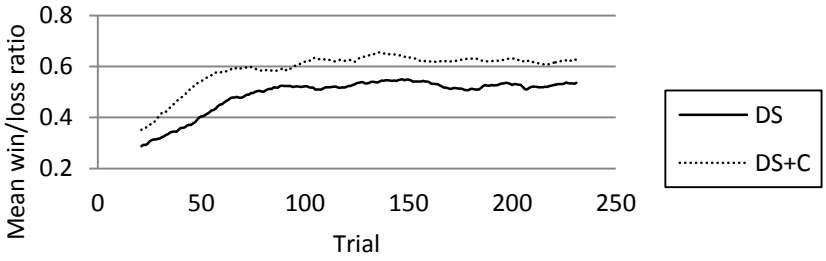

Fig. 4. Rolling mean (window size twenty) of win/loss ratio of the 'blues' against mixed tactics, with DS and DS+C. Ratios are averaged over one hundred learning episodes.

Table 1 shows the TPs of DS and DS+C against the mixed tactics. DS+C agents generally reached all TPs (50%, 60%, 70%, 80% wins) earlier than DS agents did. Note that the standard deviation in TP generally has the same order of magnitude as its mean. Independent two-sample two-tailed t-tests show that against the mixed tactics at TP(50%), the mean TPs are achieved significantly earlier using DS+C ($t = 3.85$, $p = 0.00016$) at the $a = 0.05$ significance level. The same holds for TP(60%) ($t = 3.60$, $p = 0.00039$), TP(70%) ($t = 3.60$, $p = 0.00039$), and TP(80%) ($t = 2.46$, $p = 0.015$).

In contrast with the performance against opponents that employed mixed tactics, TPs for DS+C agents were generally achieved later against the basic tactics. The learning curve of DS and DS+C against the mixed tactics is shown in Fig. 4. Both DS and DS+C agents seem to have passed a point of inflection after around 100 trials. After the first 100 trials, DS and DS+C maintain a mean win/loss ratio of 0.53 and 0.63, respectively. The mean percentage difference between the learning curves is 20.3%, with DS+C agents clearly outperforming DS agents during the entire learning process.

Log traces show that the coordination rules were selected and activated multiple times. This means that according to the DS algorithm, the coordination rules had added value. Considering the final weights of the rules, it can be observed that some of the coordination rules received high weights. The 'blue lead' favored one rule in particular, with a mean final weight of 178.6. This rule stated 'if I receive a message that my wingman is evading an enemy, turn approximately towards the enemy'. Interest-

ingly, the ‘blue wingman’ mainly favored two rules, with mean final weights of 103.8 and 106.6. Both rules made the ‘wingman’ perform an evasive action when it received a message that the ‘lead’ was trying to avoid being detected by ‘red’.

In the case without coordination, the rule that stood out most was the so-called ‘beam’ maneuver (flying perpendicular to an enemy’s radar to avoid detection). This rule received high weights from the ‘blue lead’ (386.7) and the ‘blue wingman’ (323.5). A final interesting observation is that in all cases the agents preferred firing from a greater distance. This had the obvious advantage that it would be hard for ‘red’ to hit a ‘blue’, but would also diminish the chances of the ‘blues’ to make a hit on red.

6 Discussion

In this paper, we have presented a method for team coordination through communication using DS, called DS+C. The method was tested in a (simulated) air combat environment, in which a team of two learning agents had to learn how to defeat an opponent. Over a large set of experiments, DS+C showed clear advantages over traditional DS for multi-agent reinforcement learning. Throughout the learning process of 250 trials (an episode) DS+C agents won more often than DS agents from opponents that frequently change their tactics. On the basis of a decentralized coordination scheme, DS+C agents are able to develop more successful and more robust tactics against a less predictable opponent. Coordination in multi-agent systems is an extensively researched topic, with many issues and learning opportunities [9]. From the literature we know that other authors have found that the addition of coordination to a multi-agent system does not automatically lead to increased performance [10].

To judge the relative efficiency of DS and DS+C, we defined the TP(X) measure, based on the TP measure from [7]. DS+C agents reached the TP(X) at 50%, 60%, 70% and 80% significantly earlier than regular DS did, against an opponent with *mixed* tactics. From the fact that DS+C reached these ‘milestones in learning’ earlier than DS did, we may conclude that DS+C agents learn more efficiently than DS agents.

Looking at the learning curves shown in Fig. 4, it can be observed that DS+C agents generally maintain a higher win/loss ratio than DS agents throughout the learning process, against opponents that employed mixed tactics. Therefore, we may provisionally conclude that in this case, DS+C agents are not only more efficient in their learning process, but also more effective than DS agents, after training.

The higher performance of DS+C should probably be attributed to the addition of more evasive rules to the rule bases. Since the ‘blues’ would lose if only a single ‘blue’ was hit, cautious behavior was rewarded. This can be seen in the high weights that several evasive rules received. Also, because the coordination rules were proven to be valuable, it is also easy to explain the faster convergence on optimal scripts, since the DS+C agents simply had more good options available. However, the coordination rules were not intentionally biased towards evasion, and it remains possible that more aggressive rules would have a similar effect.

Against the opponent with a basic tactic, the picture is slightly different. As can be seen in Table 1, DS+C agents also reached the TPs earlier against opponents with a basic tactic. However, the TPs against the basic tactics were achieved relatively early for both DS and DS+C. Surprisingly, against the *Close Range* tactic, DS achieved earlier TPs than DS+C did. We hypothesize that if DS was able to rapidly find optimal behavior against this tactic of 'red', then the additionally included coordination rules for DS+C only hindered the convergence to successful rules, resulting in later TPs. Additionally, there seemed to be a trend of both DS and DS+C having later TPs against the alternative (reverse direction) versions of tactics (see Section 4). Additional experiments, in which the formation of the 'blues' was mirrored, also led to later TPs, when the opponent employed a non-reversed tactic. While this can be considered an artefact, it is also an indication that the spatial configuration of a formation of cooperating aircraft is a relevant factor in air combat.

Table 1 shows that the means and standard deviations of the TPs generally had the same order of magnitude. Each learning episode starts with a rule base in which each rule has an equal weight. There are nevertheless two sources of variance when averaging TPs over episodes. The first source is the stochastic sampling of the rule base by the DS algorithm. The second source is stochastic variation in the simulation environment (e.g., radar detection probability and missile kill probability). These sources cause stochastic variations in win/loss ratio and hence stochastic differences between episodes. Note that the first source of variance is non-stationary, in the sense that the distribution of weights in the rule base continuously changes during an individual episode, and eventually diminishes after a subset of relatively successful rules are identified by the algorithm.

The high weights of both general and coordination rules promoting 'evasion' were likely caused by the fact that 'blue' would lose the trial if only one of the two 'blues' was hit. Thus, 'blue' was relatively vulnerable. At the same time, the two 'blues' together had more missiles at their disposal than red, thereby promoting the 'distant firing' rules as well, overall resulting in a low risk strategy.

7 Conclusions and Future Work

From the experimental results given above we may conclude that the difference in performance against mixed tactics is the most interesting outcome: it shows that DS+C agents are better able to generalize their behavior against unpredictable enemies than DS agents.

The next step is to expand the scenario and investigate the use of DS+C with more agents, both friendly and enemy. Further work could investigate how existing extensions to DS, such as performance enhancements [7] and extensions leading to variety in the learned behavior [11] would interact with DS+C. In the future it could also be investigated which communication is most effective against specific enemy tactics, or if a centralized coordination method would offer any benefits over the currently used decentralized method. Research in these directions will further improve CGF behavior and the effectiveness of training simulations.

Acknowledgments

LtCol Roel Rijken (Royal Netherlands Air Force) provided the first version of the simulation environment used in this work. The authors also thank Pieter Huibers and Xander Wilcke for their assistance with the simulation environment.

References

1. Laird, J.E.: An exploration into computer games and computer generated forces. Eighth Conference on Computer Generated Forces and Behavior Representation (2000).
2. Fletcher, J.D.: Education and training technology in the military. Science. 323, 72–5 (2009).
3. Roessingh, J.J., Merk, R.-J., Huibers, P., Meiland, R., Rijken, R.: Smart Bandits in air-to-air combat training: Combining different behavioural models in a common architecture. 21st Annual Conference on Behavior Representation in Modeling and Simulation. , Amelia Island, Florida, USA (2012).
4. Benjamin, P., Graul, M., Akella, K.: Towards Adaptive Scenario Management (ASM). The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC). pp. 1478–1487. National Training Systems Association (2012).
5. De Kraker, K.J., Kerbusch, P., Borgers, E.: Re-usable behavior specifications for tactical doctrine. Proceedings of the 18th conference on behavior representation in modeling and simulation (BRIMS 2009). pp. 15–22. , Sundance, Utah, USA (2009).
6. Koopmanschap, R., Hoogendoorn, M., Roessingh, J.J.: Learning Parameters for a Cognitive Model on Situation Awareness. The 26th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems. pp. 22–32. , Amsterdam, the Netherlands (2013).
7. Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I., Postma, E.: Adaptive game AI with dynamic scripting. Mach. Learn. 63, 217–248 (2006).
8. Van der Sterren, W.: Squad Tactics: Team AI and Emergent Maneuvers. In: Rabin, S. (ed.) AI Game Programming Wisdom. pp. 233–246. Charles River Media, Inc. (2002).
9. Stone, P., Veloso, M.: Multiagent systems: A survey from a machine learning perspective. Auton. Robots. 8, 345–383 (2000).
10. Balch, T., Arkin, R.C.: Communication in reactive multiagent robotic systems. Auton. Robots. 1, 27–52 (1994).
11. Szita, I., Ponsen, M., Spronck, P.: Effective and Diverse Adaptive Game AI. IEEE Trans. Comput. Intell. AI Games. 1, 16–27 (2009).