Rewarding Air Combat Behavior in Training Simulations

Armon Toubman,
Jan Joris Roessingh
Department of Training, Simulation, and Operator Performance
National Aerospace Laboratory NLR
Amsterdam, Netherlands
{Armon.Toubman, Jan.Joris.Roessingh}@nlr.nl

Pieter Spronck
Tilburg center for Cognition and Communication
Tilburg University
Tilburg, Netherlands
p.spronck@gmail.com

Aske Plaat,
Jaap van den Herik
Leiden Institute of Advanced Computer Science
Leiden University
Leiden, Netherlands
{aske.plaat, jaapvandenherik}@gmail.com

***Abstract*—Computer generated forces (CGFs) inhabiting air combat training simulations must show realistic and adaptive behavior to effectively perform their roles as allies and adversaries. In earlier work, behavior for these CGFs was successfully generated using reinforcement learning. However, due to missile hits being subject to chance (a.k.a. the *probability-of-kill*), the CGFs have in certain cases been improperly rewarded and punished. We surmise that taking this probability-of-kill into account in the reward function will improve performance. To remedy the false rewards and punishments, a new reward function is proposed that rewards agents based on the expected outcome of their actions. Tests show that the use of this function significantly increases the performance of the CGFs in various scenarios, compared to the previous reward function and a naïve baseline. Based on the results, the new reward function allows the CGFs to generate more intelligent behavior, which enables better training simulations.**

Keywords—reinforcement learning; rewards; air combat; training simulations; computer generated forces

I. INTRODUCTION

Air combat training simulations require realistic and adaptive computer generated forces (CGFs) for optimal training efficacy. Designing effective behavior for these CGFs is a hard problem, as they must be able to deal with a range of possible situations. Ideally, the generation of CGF behavior is automated, in an effort to lighten the workload of training scenario developers. Reinforcement learning (RL) methods, which allow the CGFs to discover the correct actions to take in their environment, may provide a solution.

In RL methods, agents learn behavior through feedback from the environment, in the form of reward signals [1]. The reward signals vary based on how the environment has changed after the actions of the agent, in relation to some task. Typically, this reward is 0 (failure) or 1 (success) [2]. However, more complex learning problems usually require more complex rewards.

Examples of domains with complex RL problems that are actively researched are robotics [3, 4] and video games [5]. Due to the complexity of the tasks in these domains it is unhelpful for learning agents to have their tasks simply labelled a 'success' or 'failure'. This is why the rewards (or rather the functions that generate them) for these learning problems are carefully crafted using a variety of techniques.

Another example of a complex RL problem domain is that of air combat, which is of particular interest to us. Air combat is a 'game' with multiple agents, limited resources, and a partially observable environment. In air combat, 'success' means hitting the enemy aircraft with a missile. Missile hits, however, are subject to chance (known as the *probability-of-kill*), even under optimal circumstances [6]. The probability-of-kill of a missile is determined by all of the factors that have an effect on the odds of the missile hitting the target, such as the missile being properly installed on its launch platform, the target deploying countermeasures or decoy targets, or the target out-maneuvering the missile. As a result, it is possible for an agent to have a 'good' (i.e. near-optimal) policy, but at the same time miss out on a reward because its missile misses the target by chance. Likewise, it is possible for agents to have a 'bad' (i.e. sub-optimal) policy, but to also hit a target against the odds. Such chance hits and misses obstruct learning: in the former case, good behavior is not rewarded, and therefore not reinforced; and in the latter, sub-optimal behavior is rewarded, steering the policy towards local optima.

Air combat missions are often flown in multiples of two. Previous research therefore focused on learning air combat behavior with team coordination. Toubman et al. investigated the use of both centralized [7] and decentralized [8] team coordination (CTC/DTC). They found that a team of two performed better with coordination than without. Also, it was found that agents with CTC performed better than with DTC. This was surprising, as it was expected that the agents with DTC would be able to come up with a more diverse array of solutions.

A downside to the CTC approach used in [7] is that it was difficult to scale to larger teams, as the coordinating agent would require initial knowledge about all teammates. A DTC scheme is therefore more desirable. We suspect that the difference in performance between CTC and DTC where CTC learned more effective behavior and learned it faster too, was due to nondeterministic factors in the environment. While agents with CTC are able to quickly find an acceptable policy, agents with DTC are still busy optimizing their policies against their enemy, but also relative to each other. Taking into account the nondeterminism (i.e. the probability-of-kill) in the rewards may possibly help steer DTC to better performance.

In this paper, we propose a reward function that rewards agents proportionally to the expected payoff of their actions (rather than the actual payoff, which as we argue is subject to random variations). To the best of our knowledge, this is the first time such a reward function has been applied in RL in the air combat domain. We report the results of applying this technique to learning agents in an air combat simulation, and compare the results with previous research.

This paper is structured as follows: Section II gives an overview of related work. Section III describes our proposed technique, and Section IV describes the experiment we used to test our technique. Section V has the results from this experiment. The paper is concluded with a discussion of the results in Section VI and some conclusions in Section VII.

II. Related Work

Designing good reward functions is hard, and various approaches have been taken to the design of reward functions. These approaches are generally found under the name of reward shaping [9, 10] or shaping reinforcement [4].

A key point of shaping techniques is using domain knowledge in the reward function. Domain knowledge can for example aid in rewarding progress on tasks [11] or dividing the main goal into subtasks and providing rewards per subtask [12, 13]. In general, rewarding agents for taking steps towards a goal, rather than only rewarding completion of the goal, has been shown to speed up learning [2].

Air combat behavior for training simulations is one of those complex domains that need a specialized reward function. The domain offers many constraints and freedoms: preparation in terms of armament and mission planning are done in advance according to well-structured guidelines, while the actual implementation of combat maneuvers is left to the fighter pilots. Agents learning air combat behavior have to deal with noisy, partially observable states, including teammates reaching for a shared goal and adversaries with opposing goals. Also, research in this domain has a substantial practical application, as well-trained agents can be used as allies and adversaries in training simulations.

A prevalent example of learning air combat behavior is [14]. Smith et al. used a learning classifier system (LCS) to automatically generate fighter jet maneuvers for *within visual range* air combat. They report that they considered a number of measures for the effectiveness of the maneuvers, before settling on a measure based on the advantage in terms of the aspect angle (i.e. the angle of the attacker's nose to the defender's tail). Situations, in which the learning agent achieved a sufficiently small aspect angle, meaning it had an opportunity to fire its gun, were awarded a higher score.

In *beyond visual range* air combat, Toubman et al. [7, 8] employed RL with a reward function that was composed of rewards for winning a trial, the duration of the trial and the use of resources. This composition, including the importance of each element, was manually designed based on domain knowledge. While these may be valid performance indicators, the importance of each of them is hard to balance in a single reward function. We believe it is this balance that creates the difference in performance between agents with CTC and DTC. Therefore, we propose a new method of rewarding agents.

III. Probability-of-kill Rewards

The success of air-to-air missiles is subject to a number of factors, such as the distance to the target, the deployment of countermeasures by the target, the maneuverability of the target, technical malfunctions, etc. Together, these factors determine the probability-of-kill (P_k) of the missile.

Combat aircraft are only able to carry a limited number of missiles. Firing a missile is therefore a risky action, in the sense that firing a missile, even in optimal circumstances, does not guarantee hitting the target. This is an obstacle for learning in repeated simulations. Actions with a high probability of success can still fail. Likewise, actions with a low probability of success can still be successful. This means that trying out a 'successful' policy can still result in punishment and 'bad' policies can collect rewards.

One possible solution to remedy this problem is to repeat simulations with the same policies, and average the outcomes. This is however inherently computationally expensive. Therefore, we are interested in a simpler solution to this problem.

We propose a reward function that rewards agents proportionally to the *expected outcome* of actions with a nondeterministic result. The agent that fired the missile with the first impact directly receives the calculated P_k as a reward. Effectively, this means that when an agent performs a risky action, that has the potential of completing the task (i.e. destroying the enemy), it is rewarded with the expected payoff of that action. The agent is not rewarded for a 'lucky' hit, nor does it miss rewards for 'unlucky' misses.

The calculated P_k of the first missile is subtracted from the maximum possible reward (starting at 1), leaving the remainder as the new maximum reward. On subsequent missile impacts, the reward is calculated in the same manner, including updating the maximum reward. Equation 1 shows this process, with *a'* being the agent that fired missile *m'*, *A* being the set of all agents and *M* being the set of all missiles.

$$\mathrm{r}(a', m') = \left[1 - \sum_{a \in A, m \in M} \mathrm{r}(a, m)\right] \cdot \mathrm{P}_\mathrm{k}(m') \qquad (1)$$

By design, this reward function will have less reward available to divide with subsequent missile impacts. As the first missile hit in an encounter has the potential to be the deciding factor in an encounter, it therefore should have the most potential reward.

IV. Experimental Setup

The *probability-of-kill* reward function was tested in an air combat simulation. The scenario used in the simulation was similar to the scenario used by Toubman et al. [7, 8]: a 2-versus-1 air combat scenario in which two blue fighter jets engage a red fighter jet that is performing a *combat air patrol* (CAP) in a section of airspace. A screenshot of the simulation is shown in Fig. 1.

The CGFs are described in Section IV-A. Sections IV-B and IV-C describe the blue team and the red team respectively. Section IV-D shows the parameters that were used for the learning process, and Section IV-E describes the reward functions that will be compared. The analysis of the experimental results is described in Section IV-F.

A. CGFs

The fighter jets in the simulation are based on the F-16, and each jet is equipped with radar and a radar warning receiver (a device that detects incoming radar signals). Furthermore, each jet carries four medium range, active radar homing missiles based on the AIM-120B AMRAAM.

Each fighter jet is controlled by an agent. During each trial, the behavior of each agent is governed by a script: the scripts of the agents in the blue team are generated by Dynamic Scripting (DS) (see Section IV-B) while the agents in the red team have a script for each tactic they use (see Section IV-C).

The behavior rules in each script are if-then rules which map observations to actions. Each time step, a matching rule is selected and executed. Examples of the rules are "if I see an enemy on my radar, and this enemy is within 80 kilometers of me, and I have missiles left, fire a missile at this enemy" and "if I detect an incoming missile, turn right 180 degrees".

B. Blue team

The blue team consists of two agents, a 'lead' and a 'wingman'. As mentioned, the blues learn using DS. DS is a reinforcement learning technique that optimizes a script (the policy) through the recombination of behavior rules [15]. An initial rule base is provided, in which each rule has an initial weight. Each rule's weight forms the probability of that rule to be included in a newly generated script (with a maximum number of rules per script). Feedback from executing the script in the environment leads to adjustments in the weights of the rules included in the script, thereby changing the odds of rules' appearances in new scripts.

To allow the blue agents to learn tactics using DS, some of the rules in their rule bases are duplicated and varied slightly (e.g. with variations in the angle of their turn or the distance at which they fire a missile). Furthermore, the blues are able to coordinate their actions through the use of a decentralized coordination scheme based on [8].

At the beginning of each trial, the blues are positioned so that they fly into red's CAP. The rule bases of the blues are mostly identical; except for the inclusion of rules in the blue wingman's rule base that allow it to fly in several formations with its wingman.

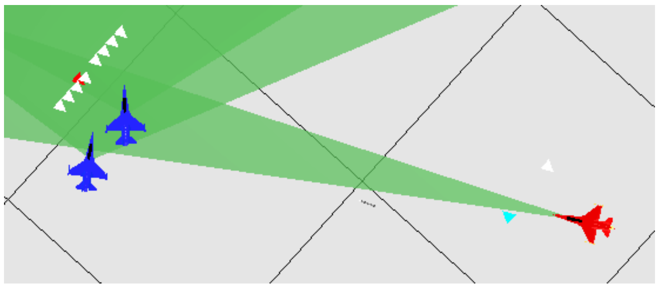
Figure 1. Screenshot of the simulation.

C. Red team

The red team consists of one agent that uses one of four scripted tactics throughout a learning episode (i.e. set of subsequent trials). This includes the following three basic tactics:

- Default tactic: initially fly a CAP pattern counterclockwise, and engage the blues upon detection;
- Evading tactic: as the default tactic, but red tries to avoid the blues' missiles;
- Close range tactic: as the default tactic, but red only fires missiles from a closer range, giving the blues less time to evade them.

The fourth tactic is a mixed tactic in which red repeatedly picks one of the three basic tactics at random and uses it until it loses, at which point it picks a new tactic. This mixed tactic is included to test the blues' capability to generalize their own behavior when given a changing problem.

D. Reward functions

The blues are rewarded during learning using one of the following three reward functions.

1) Binary rewards

The blues are rewarded with 1 if they won the trial and 0 if they lost. This reward function provides the baseline.

2) Domain knowledge-based rewards

This reward function, used previously in [7, 8], uses domain knowledge to judge the behavior produced by the blue agents. The function consisted of 0 or 1 if the team respectively lost or won the trial (weighing for 3/4 of the total reward), ratio of the time taken to the decisive missile impact to the maximum allowed duration of a trial (weighing 1/8), and a score for the amount of resources used in the trial (also weighing 1/8) (currently, for the blue team this is the ratio of missiles left to the number of starting missiles minus one if the blue team won, and one minus the ratio of missiles left to the number of starting missiles if the blue team lost).

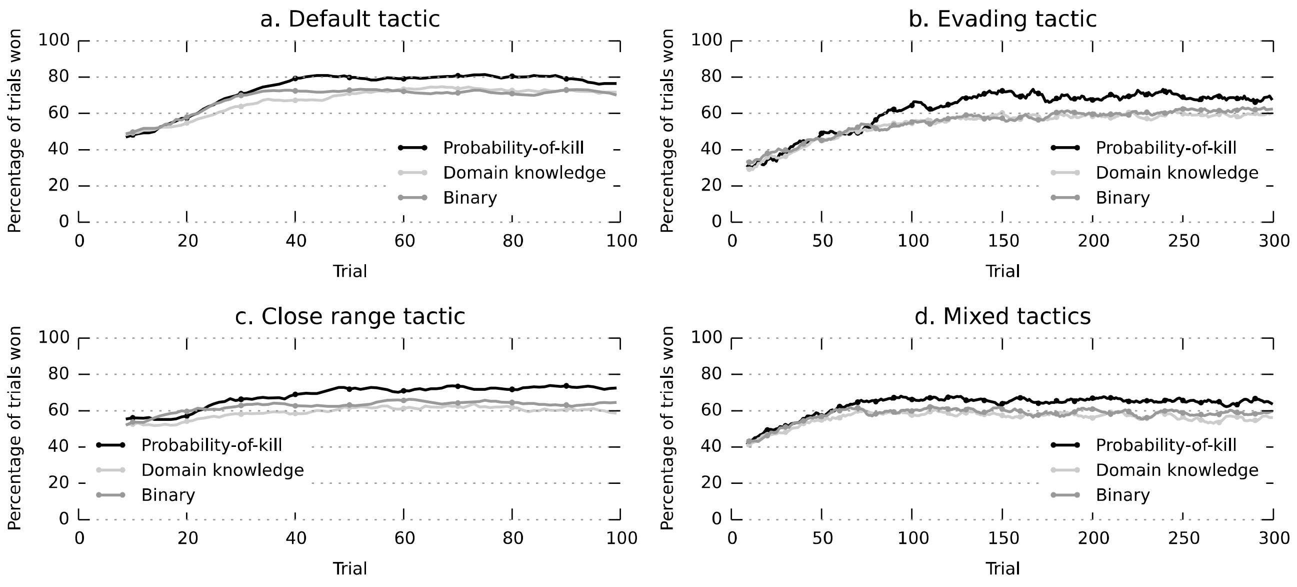

Figure 2. Percentages of trials won by the blue team using each of the three reward functions (probability-of-kill, domain knowledge-based and binary), against each of red's tactics. Rolling mean, window size 10.

3) *Probability-of-kill rewards*

The blues are rewarded with the P_k reward function described in Section III.

It should be noted that the three reward functions listed here reward the agents for optimal behavior, rather than realistic behavior. While realistic behavior is needed for actual training scenarios, optimal behavior is easier to quantify and therefore more suitable to use for the comparison between the reward functions.

E. Learning parameters

A learning episode consists of 100 consecutive trials. Against each of red's tactics, 150 learning episodes are performed using each of the three reward functions.

To facilitate learning using the P_k reward function, the missiles carried by the jets are made harmless, in the sense that they calculate their P_k on impact but do not actually kill the target. The P_k model of each missile is a linear function of the distance flown by the missile from its launch to impact, shown in Equation 2.

$$P_k(m) = 80 - \frac{\text{distance flown by m (km)}}{80} \quad (2)$$

Each missile determines whether it would have hit the target, using its P_k. The team that scores the first hit in a trial is said to have won that trial.

F. Hypotheses and analysis of results

We predict that using the P_k reward function to reward the learning agents will result in a higher percentage of won trials after learning (i.e. higher effectiveness). We also predict that the better performance will be noticeable throughout the learning process (i.e. higher efficiency).

Each trial, we record whether the blue team won or lost. This procedure yields 150 win/loss sequences, using each of the three reward functions, against reach of red's four tactics.

Analysis is done using the randomized ANOVA method presented in [16], which was designed for the comparison of machine learning performance curves. This method uses repeated ANOVAs to compensate to the carryover effect in the data, as time spent learning has an effect on the performance. However, as our performance data consists of sequences of zeroes and ones, the data is first transformed by taking the average of groups of 30 sequences. This results in 5 average sequences per configuration, which are used as input for the ANOVAs.

Furthermore, we inspect at which point the performance stabilizes and the agents do not benefit from further learning.

V. Results

For each combination of red's tactics against the blues' coordination methods, the blues' rewards were collected over 150 learning episodes consisting of 100 trials. However, as the performance did not level off after 100 trials against the evading and the mixed tactics, the blues were allowed to continue learning for 300 trials in these cases. Fig. 2 shows the learning curves in terms of the ratio of trials won by blue against each of red's tactics.

The learning curves were submitted to a two-way ANOVA, with three levels of reward function and 100/300 levels of training (i.e. amount of trials). This was done for each of red's tactics. The significance of the main and interaction effects were calculated using the method from [16]. The resulting values are listed in Table I.

Table II shows the trials at which the performance stabilized, as visualized in Fig. 2. The remainder of the performance curve of P_k rewards was compared to those of the

TABLE I. ANOVA FOR EACH TACTIC OF RED

Tactic	Effect		*F*	*p*
Default	Rewards	*F*(2,1200)	80.781	* .002
	Interaction	*F*(198,1200)	1.234	* .005
Evading	Rewards	*F*(2,3600)	579.981	* .005
	Interaction	*F*(598,1200)	1.525	* .005
Close range	Rewards	*F*(2,1200)	132.966	* .001
	Interaction	*F*(198,1200)	1.166	* .002
Mixed	Rewards	*F*(2,3600)	256.318	* .004
	Interaction	*F*(598,3600)	0.969	.077

* significant at the $a = 0.05$ level

TABLE II. MEAN PERCENTAGE DIFFERENCE IN PERFORMANCE AFTER STABILIZATION

Tactic	Stabilized at trial	Mean percentage difference	
		P_k / Domain Knowledge	P_k / Binary
Default	40	+ 10.0%	+ 10.9%
Evading	150	+ 18.1%	+ 15.0%
Close range	50	+ 19.4%	+ 12.4%
Mixed	100	+ 15.1%	+ 11.0%

domain knowledge-based reward function and the binary reward. After learning, the blues performed between 10 and 19.4 percent better using the P_k reward function.

VI. DISCUSSION

Using the P_k reward function provides a clear advantage over the domain knowledge-based reward function and the binary reward function, as can be seen in Fig. 2. The results listed in Table I corroborate this: significant effects of the reward function were found in all cases. Additionally, the significant interaction effects show that in three out of four cases, the relationship between training and performance depends on the reward functions. Only against the mixed tactics does the interaction effect appear to be non-significant by our measure, which can be explained as sampling error in the randomized ANOVA analysis.

Fig. 2 shows that the performance of the blues using the P_k rewards against the default tactic and the close range tactic stabilizes around 40 and 50 trials respectively. As the agents were not done learning after 100 trials against the evading and mixed tactics (i.e. their performance had not yet stabilized), they were allowed to keep learning to 300 trials. Further learning only affected the performance as the evading tactic, which continued to climb up to trial 150. On the other hand, the performance against the mixed tactic leveled off near trial 100.

Table II also shows the actual increase in performance of using P_k rewards against the other reward functions. In all cases, the blues were able to reach the highest performance using the P_k rewards. On average, the increases over the domain knowledge-based reward function are larger than the increases over the binary reward function. This is interesting, as it shows that the addition of domain knowledge in its current form negatively affects performance compared to simply rewarding wins and punishing losses.

Out of the four tactics, the blues take the longest to learn optimal behavior against the evading tactic. Also, the behavior that is learned against the mixed tactic is the worst, with the lowest mean performance. The performance against the mixed tactic is expected, as the changing problem is hard to optimize against. However, stabilizing at around 65%, the performance is only slightly worse than the performance against the separate tactics.

The question may arise why the performance levels off below 100 percent at all. This is due to three factors. First, because of the *probability-of-kill* discussed in this paper: similar trials may end up differently because the missile hits provide different outcomes and different winners. Secondly, the DS algorithm creates scripts in a stochastic manner, possibly leading agents to different optima each time. Finally, DS does not synthesize new behavior. Instead it recombines pre-written behavior rules. As a result, if there is no absolute optimal combination of rules present in the rule base, optimal behavior is not possible.

It is interesting to see the blues' performance against the evading tactic climb from around 30% to over 70%. The blues are able to optimize their performance the most against this tactic, although slowly. As the red agent actively tries to evade the blues' missiles, the lucky hits and unlucky misses described earlier start to matter, as the team that scored the first missile hit in a trial won that trial. It is here that the P_k reward function really shows its benefits (see Fig. 2b), as it allows the blues to keep gradually improving their shots.

The P_k model was of course highly abstracted. In reality, the *probability-of-kill* of a missile is much more complicated. We believe our model helps show the principle of the P_k rewards. However, for actual training applications, the agents would have to be able to cope with more complex factors. It would be interesting to see if the improved performance using the P_k rewards carries over to a scenario with noisy data, for example.

VII. CONCLUSIONS

By rewarding learning air combat agents with the *probability-of-kill* of the missiles they fire, they are no longer falsely rewarded for missiles that hit by chance, or falsely punished for missiles that missed by chance. These rewards resulted in moderate performance increases (10 to 19.4%) over methods used in earlier work, in various scenarios. In essence, the use of the P_k rewards allows us to more effectively generate air combat behavior for CGFs. These CGFs are able to learn task solving behavior, despite nondeterministic events in their environment. This, in turn, will lead to easier and better development of CGFs for training scenarios. In essence, the P_k reward function enables a more human-centered design of the scenario building process.

The application of the P_k rewards is not limited to air combat simulations. The concept naturally extends to land-based, naval, and space combat simulations. We also envision usage in safety applications, where agents must cope with systems composed of components with some failure rate.

The reward function proposed in this paper may also be relevant for scoring human behavior in air combat training. In any case, both human and virtual agents may benefit from a more realistic P_k model. Further research is needed to investigate the effects of a more complex P_k model on the performance of agents, as well as the integration of multiple P_k models for different weapons.

Acknowledgment

The authors thank Lt Col Roel Rijken (Royal Netherlands Air Force) for the first version of the simulation environment and advice regarding air combat tactics, and Pieter Huibers and Xander Wilcke for further work on the simulation.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*: MIT Press, 1998.

[2] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," *J. Artificial Intell. Res.,* vol. 4, pp. 237-285, 1996.

[3] J. Kober, D. Bagnell, and J. Peters, "Reinforcement Learning in Robotics: A Survey," *Int. J. of Robotics Res.,* pp. 1238-1274, 2013.

[4] M. J. Matarić, "Reinforcement Learning in the Multi-Robot Domain," *Auton. Robots,* vol. 4, pp. 73-83, 1997.

[5] I. Szita, "Reinforcement Learning in Games," in *Reinforcement Learning*. vol. 12, M. Wiering and M. van Otterlo, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 539-577.

[6] M. R. MacLeod, "Preventing Premature Conclusions: Analysis of Human-In-the-Loop Air Combat Simulations," in *29th Int. Symp. Military Oper. Res.*, 2012.

[7] A. Toubman, J. J. Roessingh, P. Spronck, A. Plaat, and J. van den Herik, "Centralized Versus Decentralized Team Coordination Using Dynamic Scripting," in *Proc. 28th Eur. Simulation and Modelling Conf.*, Porto, Portugal, 2014, pp. 129-134.

[8] A. Toubman, J. J. Roessingh, P. Spronck, A. Plaat, and J. van den Herik, "Dynamic Scripting with Team Coordination in Air Combat Simulation," in *Proc. 27th Int. Conf. Ind. Eng. and Other Applicat. of Appl. Intell. Syst.*, Kaohsiung, Taiwan, 2014, pp. 440-449.

[9] A. Laud and G. DeJong, "The Influence of Reward on the Speed of Reinforcement Learning: An Analysis of Shaping," in *Proc. 20th Int. Conf. Mach. Learning*, Washington, DC, 2003, pp. 440-447.

[10] A. Y. Ng, D. Harada, and S. J. Russell, "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping," in *Proc. 16th Int. Conf. Mach. Learning*, Bled, Slovenia, 1999, pp. 278-287.

[11] M. J. Mataric, "Reward Functions for Accelerated Learning," in *Proc. 11th Int. Conf. Mach. Learning*, New Brunswick, NJ, 1994, pp. 181-189.

[12] S. Mahadevan and J. Connell, "Scaling reinforcement learning to robotics by exploiting the subsumption architecture," in *Proc. 8th Int. Workshop on Mach. Learning*, Evanston, IL, 1991, pp. 328-332.

[13] J. Randlov and P. Alstrom, "Learning to drive a bicycle using reinforcement learning and shaping," in *Proc. 15th Int. Conf. Mach. Learning*, Madison, WI, 1998, pp. 463-471.

[14] R. E. Smith, B. A. Dike, R. K. Mehra, B. Ravichandran, and A. El-Fallah, "Classifier systems in combat: two-sided learning of maneuvers for advanced fighter aircraft," *Comput. Methods Appl. Mechanics and Eng.,* vol. 186, pp. 421 - 437, 2000.

[15] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma, "Adaptive game AI with dynamic scripting," *Mach. Learning,* vol. 63, pp. 217-248, 2006.

[16] J. H. Piater, P. R. Cohen, X. Zhang, and M. Atighetchi, "A Randomized ANOVA Procedure for Comparing Performance Curves," in *Proc. 15th Int. Conf. Mach. Learning*, Madison, WI, 1998, pp. 430-438.