



NLR-TP-2013-507

## **Dynamic Scripting with Team Coordination in Air Combat Simulation**

A. Toubman, J.J.M. Roessingh, P. Spronck, A. Plaat and  
H.J. van den Herik

**Nationaal Lucht- en Ruimtevaartlaboratorium**

National Aerospace Laboratory NLR

Anthony Fokkerweg 2

P.O. Box 90502

1006 BM Amsterdam

The Netherlands

Telephone +31 (0)88 511 31 13

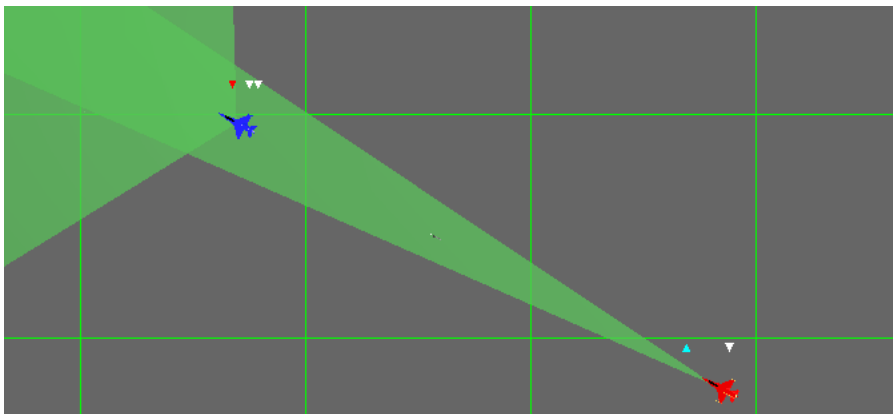
Fax +31 (0)88 511 32 10

[www.nlr.nl](http://www.nlr.nl)



## Executive summary

# Dynamic Scripting with Team Coordination in Air Combat Simulation



### Report no.

NLR-TP-2013-507

### Author(s)

A. Toubman  
J.J.M. Roessingh  
P. Spronck  
A. Plaat  
H.J. van den Herik

### Report classification

UNCLASSIFIED

### Date

January 2014

### Knowledge area(s)

Training, Missiesimulatie en Operator Performance

### Descriptor(s)

Machine Learning  
Multi-Agent Systems  
Autonomous Agents  
Computer Generated Forces  
Air Combat

### Problem area

In training simulations, Computer Generated Forces (CGFs) play the role of enemy or teammate. Traditionally, behavior of CGFs is controlled through scripts. Building such scripts requires time and expertise. The process of building scripts can possibly be automated using Machine Learning techniques.

### Description of work

We have created an extension to the Dynamic Scripting (DS) technique with team coordination, called DS+C. In DS+C, team members exchange information about their current state. We applied DS+C in a 2v1 air combat simulation. In the simulation, two 'blues' used DS+C to learn team combat behavior against a 'red' enemy. The 'red' agent used various tactics.

### Results and conclusions

Against an unpredictable enemy using a mix of tactics, a 20% performance increase was obtained using DS+C over regular DS. Also, with DS+C, effective behavior was learned faster. We conclude that DS+C is a valuable extension to DS when learning behavior for CGFs in training simulations, which will fight unpredictable human adversaries.

### Applicability

DS+C can be used to generate more effective CGF behavior for use in team-based simulations. Because of its robustness against varying enemy tactics, DS+C is particularly applicable for the generation of behavior for use in training simulations.

This report is based on a paper to be published at the 27th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, Kaohsiung, Taiwan, June 3-6, 2014.





NLR-TP-2013-507

## Dynamic Scripting with Team Coordination in Air Combat Simulation

A. Toubman, J.J.M. Roessingh, P. Spronck<sup>1</sup>, A. Plaat<sup>1</sup> and  
H.J. van den Herik<sup>1</sup>




<sup>1</sup> Tilburg University

This report is based on a paper to be published at the 27th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, Kaohsiung, Taiwan, June 3-6, 2014.

The contents of this report may be cited on condition that full credit is given to NLR and the authors.  
This publication has been refereed by the Advisory Committee AIR TRANSPORT.

Customer	National Aerospace Laboratory NLR
Contract number	---
Owner	NLR
Division NLR	Air Transport
Distribution	Unlimited
Classification of title	Unclassified
	January 2014

Approved by:

Author Armon Toubman 	Reviewer Pieter Huibers 	Managing department Harrie Bohnen 
Date: 20-1-2014	Date: 20-1-2014	Date: 20-1-2014



## Summary

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 enemy. 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 under regular DS.



## **Contents**

<b>1</b>	<b>Introduction</b>	<b>5</b>
<b>2</b>	<b>Dynamic Scripting Method and Related Work</b>	<b>6</b>
<b>3</b>	<b>Dynamic Scripting with Team Coordination</b>	<b>7</b>
<b>4</b>	<b>Case Study and Experimental Setup</b>	<b>8</b>
<b>5</b>	<b>Results</b>	<b>10</b>
<b>6</b>	<b>Discussion</b>	<b>11</b>
<b>7</b>	<b>Conclusions and Future Work</b>	<b>13</b>
	<b>Acknowledgements</b>	<b>13</b>
	<b>References</b>	<b>14</b>
	<b>Appendix A Rule bases</b>	<b>15</b>



## **Abbreviations**

CGF – Computer Generated Forces

DS – Dynamic Scripting

DS+C – Dynamic Scripting with Coordination

RWR – Radar Warning Receiver





## 1 Introduction

Military organizations are increasingly using simulations for training purposes. Simulations are less costly, safer, and more flexible than training with real equipment in real-life situations (Fletcher, 2009; Laird, 2000). In military simulations, the roles of allies and adversaries are performed by computer generated forces (CGFs).

Traditionally, the behavior of CGFs is scripted (Roessingh, Merk, Huibers, Meiland, & Rijken, 2012). Production rules—rules that define actions when certain conditions hold—are manually crafted to suit specific (types of) CGFs. In complex domains, such as that of air combat, this leads to complex scripts (collections of production rules) and requires availability of domain expertise. These scripts then produce rigid behavior, because it is impossible to account for all situations that the CGFs will encounter during simulations.

Artificial Intelligence techniques may provide a solution by automating the process of generating CGF behavior, bypassing the requirement of expertise availability and shortening the time needed to generate the behavior. Various efforts have been made at realizing automatic generation of CGF behavior (also in the tangent area of scenario generation) (Benjamin, Graul, & Akella, 2012; de Kraker, Kerbusch, & Borgers, 2009).

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 (Roessingh et al., 2012), (Koopmanschap, Hoogendoorn, & Roessingh, 2013). In this paper, we diverge from the earlier approach of using cognitive models by applying ML directly to the generation of behavior.

The new envisaged 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 (Roessingh et al., 2012), (Koopmanschap et al., 2013) 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 a 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 (Spronck, Ponsen, Sprinkhuizen-Kuyper, & Postma, 2006). 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 called DS+C for 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. (Spronck et al., 2006) to address certain requirements for adaptive game AI in commercial video games, such as “easily interpretable results” and “reasonably successful behavior at all times.”<sup>1)</sup> These requirements are also applicable in the domain of military training, where quality controls such as transparent results and robust behavior are important.

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 (Spronck et al., 2006) team behavior was a result of emergence, guided by the fitness function which rewarded team victories as well as individual success. For air combat training simulations we remark that they require more control over the team

---

<sup>1)</sup> A complete description of these requirements can be found in (Spronck et al., 2006).



members' actions, to make sure that the CGFs act conforming to the training goals of a particular simulation. Such an intensive control can be formalized by coordination rules.

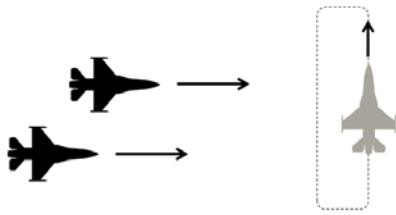
There are two methods of team coordination: centralized and decentralized coordination (van der Sterren, 2002). With centralized coordination, one agent may direct the actions of a team. With decentralized coordination, all agents in a team may influence each other's actions by sharing information through some form of communication.

Both coordination methods are candidates to be implemented using DS. In this paper, we have chosen to implement decentralized control, because its implementation is straightforward and directly understandable. Moreover, the smallest step from having a team of one agent with one 'team rule base' is an increase to a team with two agents with two nearly similar rule bases. Doctrine and mission goals are easily expressed in DS+C through rules in each rule base. To enforce the decentralized coordination, some form of communication would have to be added to the agents. However, adding communication to multi-agent systems in general is not trivial (Stone & Veloso, 2000).

### **3 Dynamic Scripting with Team Coordination**

We implemented team coordination in a DS+C environment by communication between two agents. The communication scheme is important for understanding the possible consequences of the actions by one of the two agents. For generality, we describe the system for more than two agents, viz.  $n$  agents with  $n$  rule bases. In brief, the system works as follows. Whenever an agent activates a rule, it sends a message describing the actions it is executing to its teammates. Each agent has rules in their rule bases that are activated when the agent receives particular messages. The description of actions should not be 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.



*Fig. 1. Diagram of the scenario used in the case study. The 'blues' (left) fly towards the 'red' (right). Red is flying a CAP.*

It must be emphasized that the form of coordination as described above is completely rule-based, and has been incorporated as such in the DS+C scheme. 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. Two 'blue' fighters (virtual pilots controlling fighter planes), i.e. a 'lead' together with its 'wingman', attempt to penetrate the enemy airspace. More particularly, the 'blue' formation seeks an engagement with a 'red' fighter that defends a volume of airspace, by 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 scenario is illustrated in Fig. 2.

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. These rules are included in every script, and their weights will not be changed by the DS+C process. These 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.<sup>2)</sup>

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 tactic (henceforth called mixed tactics) ‘red’ randomly selects one of the six single 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 with the least number of missiles). The fitness function is used to calculate the adjustments to the weights as follows:

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

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, therefore a larger number of scripts (combinations of rules) is possible. This would lead to more trials needed to converge to successful behavior. Since additional rules provide more options to the agents, there are also more possibilities to rapidly find optimal behavior. 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)

---

<sup>2)</sup> An overview of the rules can be found in Appendix A.



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 (Spronck et al., 2006)) 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 single 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. 3) 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 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 showed 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  $\alpha = 0.05$  significance level. The

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

Tactics of 'red'	DS	TP(50%)		TP(60%)		TP(70%)		TP(80%)	
		$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$	$\mu$	$\sigma$
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
Single (combined)	DS	55.8	56.7	66.1	57.8	87.3	66.5	122.4	82
Single (combined)	DS+C	34.8	31.3	48	42.7	65	61.3	90.7	80.6

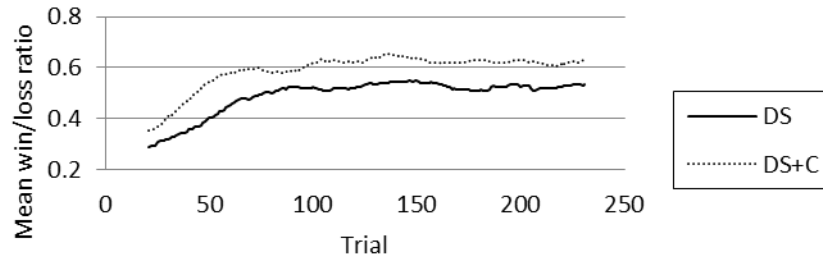


Fig. 2. Rolling mean (window size twenty) of win/loss ratio of the 'blues' against mixed tactics, with DS and DS+C. Ratios are averaged over one hundred learning episodes.

same held 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 single tactics. The learning curve of DS and DS+C against the mixed tactics is shown in Fig. 3. 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.

## 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 win 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 (Stone & Veloso, 2000). 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 (Balch & Arkin, 1994).

To judge the relative efficiency of DS and DS+C, we defined the TP(X) measure, based on the TP measure from (Spronck et al., 2006). 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. 3, 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 single 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 single tactic. However, the TPs against the single 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. Hence, ‘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 (Spronck et al., 2006) and extensions leading to variety in the learned behavior (Szita, Ponsen, & Spronck, 2009) would interact with DS+C. Future work could also investigate which communication is most effective against specific enemy tactics. In this way, CGF behavior can be tailored to more effective training simulations.

## **Acknowledgements**

LtCol Roel Rijken (Royal Netherlands Air Force) provided a 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

- Balch, T., & Arkin, R. C. (1994). Communication in reactive multiagent robotic systems. *Autonomous Robots*, *1*(1), 27–52. doi:10.1007/BF00735341
- Benjamin, P., Graul, M., & Akella, K. (2012). Towards Adaptive Scenario Management (ASM). In *The Interservice/Industry Training, Simulation & Education Conference (IITSEC)* (pp. 1478–1487). National Training Systems Association. Retrieved from <http://ntsa.metapress.com/index/T802144183868214.pdf>
- De Kraker, K. J., Kerbusch, P., & Borgers, E. (2009). Re-usable behavior specifications for tactical doctrine. In *Proceedings of the 18th conference on behavior representation in modeling and simulation (BRIMS 2009)* (pp. 15–22). Sundance, Utah, USA. Retrieved from <http://cc.ist.psu.edu/BRIMS2013/archives/2009/>
- Fletcher, J. D. (2009). Education and training technology in the military. *Science (New York, N.Y.)*, *323*(5910), 72–5. doi:10.1126/science.1167778
- Koopmanschap, R., Hoogendoorn, M., & Roessingh, J. J. (2013). Learning Parameters for a Cognitive Model on Situation Awareness. In *The 26th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems* (pp. 22–32). Amsterdam, the Netherlands
- Laird, J. E. (2000). An exploration into computer games and computer generated forces. In *Eighth Conference on Computer Generated Forces and Behavior Representation*
- Roessingh, J. J., Merk, R.-J., Huibers, P., Meiland, R., & Rijken, R. (2012). Smart Bandits in air-to-air combat training: Combining different behavioural models in a common architecture. In *21st Annual Conference on Behavior Representation in Modeling and Simulation*. Amelia Island, Florida, USA
- Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I., & Postma, E. (2006). Adaptive game AI with dynamic scripting. *Machine Learning*, *63*(3), 217–248. doi:10.1007/s10994-006-6205-6
- Stone, P., & Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, *8*(3), 345–383
- Szita, I., Ponsen, M., & Spronck, P. (2009). Effective and Diverse Adaptive Game AI. *IEEE Transactions on Computational Intelligence and AI in Games*, *1*(1), 16–27. doi:10.1109/TCIAIG.2009.2018706
- Van der Sterren, W. (2002). Squad Tactics: Team AI and Emergent Maneuvers. In S. Rabin (Ed.), *AI Game Programming Wisdom* (pp. 233–246). Charles River Media, Inc.



## **Appendix A Rule bases**

The tables below contain information on the rules that were used in the simulations.

The rules are shown together with their priority values. In Dynamic Scripting, the priority values of rules are used to break ties in case multiple rules fire during a single decision step. In such a case, only the rule with the highest priority value is allowed to fire. In case there is a tie regarding the priority values, the rule with the highest weight is allowed to fire. If the weights are also tied, a randomly chosen rule is allowed to fire.

Clarification on the radar modes: The radars used by the aircraft had two modes of operation. ‘Search’ mode produced a wide radar beam with a 120 degree angle, useful for observing a large cone in front of the area. ‘Lock’ mode produced a narrow beam with a 10 degree angle. In reality, the latter mode is used to focus all of the radar energy on a single target for tracking.



Table 2. Rules used by 'blue' with their priorities and textual descriptions.

<b>Rule name</b>	<b>Pr.</b>	<b>Description</b>
default-alone	1	If my teammate is dead, and I am 20 units away from the target area, then fly to the target area
default-fire	2	If my radar is in 'lock' mode and I have an enemy on my radar and there is no missile flying at that enemy and I have missiles left and I am within 50 units from the enemy, then fire a missile.
default-formation	1	If my teammate is alive, then fly in a '2-ship element' formation.
default-lock	2	If I have an enemy on my radar, set my radar to 'lock' mode.
default-search	2	If my radar is not in 'search' mode and I have no enemies on my radar, set my radar to 'search' mode.
default-support	2	If I have an enemy on my radar and there is a missile flying at that enemy, set my radar to 'lock' mode and fly towards that enemy.
default-target	1	If I am 20 units away from the target area, then fly to the target area.
engageRWR	6	If my Radar Warning Receiver detects an enemy radar, then turn approximately towards that radar.
evadeMissile+90	6	If I detect an incoming missile, then turn 90 degrees.
evadeMissile-90	6	If I detect an incoming missile, then turn -90 degrees.
evadeMissile180	6	If I detect an incoming missile, then turn 180 degrees.
evadeRWR+90	6	If my RWR detects an enemy radar, then turn 90 degrees.
evadeRWR-90	6	If my RWR detects an enemy radar, then turn -90 degrees.
evadeRWR180	6	If my RWR detects an enemy radar, then turn 180 degrees.
filler1 through filler5	1	If some impossible condition holds, then do nothing.
fireFrom50	9	If my radar is in 'lock' mode and I have missiles left and I am within 50 units from the enemy, then fire a missile.
fireFrom60	9	If my radar is in 'lock' mode and I have missiles left and I am within 60 units from the enemy, then fire a missile.
fireFrom70	9	If my radar is in 'lock' mode and I have missiles left and I am within 70 units from the enemy, then fire a missile.
fireFrom80	9	If my radar is in 'lock' mode and I have missiles left and



fireFrom90	9	I am within 80 units from the enemy, then fire a missile.
		If my radar is in 'lock' mode and I have missiles left and I am within 90 units from the enemy, then fire a missile.
c:filler1 through c:filler5	1	If some impossible condition holds, do nothing.
c:engaging->engage	6	If I receive a message that my teammate is turning on 'lock' mode, or is firing a missile, or is supporting a missile, and I have no enemy on my radar, and my RWR does not detect an enemy radar, move approximately to the direction of the enemy that my teammate is engaging.
c:evadeRWR->evade+90	6	If I receive a message that my teammate is evading the enemy's radar, change my heading 90 degrees plus the approximated relative bearing to the nearest enemy radar that is detected by my RWR
c:evadeRWR->evade-90	6	If I receive a message that my teammate is evading the enemy's radar, change my heading minus 90 degrees plus the approximated relative bearing to the nearest enemy radar that is detected by my RWR
c:evadeRWR->evade180	6	If I receive a message that my teammate is evading the enemy's radar, change my heading 180 degrees plus the approximated relative bearing to the nearest enemy radar that is detected by my RWR
c:evading->engage	6	If I receive a message that my teammate is evading the enemy's radar or my teammate is evading an incoming missile and I have no enemy on my radar and my RWR does not detect an enemy radar, move approximately towards the direction of the enemy that my teammate is evading.
support	5	If I have an enemy on my radar, and there is a missile flying at that enemy, then keep my radar lock on that enemy, and turn towards that enemy.
support-left	5	If I have an enemy on my radar, and there is a missile flying at that enemy, then keep my radar lock on that enemy, and turn towards that enemy minus 45 degrees.
support-right	5	If I have an enemy on my radar, and there is a missile flying at that enemy, then keep my radar lock on that enemy, and turn towards that enemy plus 45 degrees.



Table 3. Rules used by 'blue', including mean final weights against 'red' using mixed tactics.

Rule name	DS		DS+C	
	Lead	Wingman	Lead	Wingman
default-alone	n/a	50.00	n/a	50.00
default-fire	50.00	50.00	50.00	50.00
default-formation	n/a	50.00	n/a	50.00
default-lock	50.00	50.00	50.00	50.00
default-search	50.00	50.00	50.00	50.00
default-support	50.00	50.00	50.00	50.00
default-target	50.00	n/a	50.00	n/a
engageRWR	5.16	63.60	62.96	20.85
evadeMissile+90	2.65	52.33	3.96	10.37
evadeMissile-90	2.81	51.28	3.96	19.33
evadeMissile180	12.48	81.79	8.74	15.36
evadeRWR+90	3.78	106.31	14.23	92.10
evadeRWR-90	386.66	123.76	323.51	163.52
evadeRWR180	8.28	117.70	15.78	154.64
filler1 through filler5	18.89	20.11	9.16	10.41
fireFrom50	11.82	16.74	4.41	16.64
fireFrom60	13.84	26.36	12.40	37.24
fireFrom70	39.69	67.10	46.08	76.73
fireFrom80	163.64	145.23	110.98	74.99
fireFrom90	232.63	182.93	173.67	79.93
c:filler1 through c:filler:5	18.89	20.11	n/a	n/a
c:engaging->engage	n/a	n/a	51.20	81.48
c:evadeRWR->evade+90	n/a	n/a	78.42	103.78
c:evadeRWR->evade-90	n/a	n/a	35.62	106.57
c:evadeRWR->evade180	n/a	n/a	93.76	96.17
c:evading->engage	n/a	n/a	178.62	70.26
support	84.41	12.00	8.52	5.50
support-left	56.70	15.21	10.21	5.83
support-right	67.62	16.44	7.97	6.22



Table 4. Rules used by 'red'.

<b>Rule name</b>	<b>Pr.</b>	<b>Description</b>
initialCAP1	1	In all cases, change my state to 'cap1', and set my radar to 'search' mode.
flyToCAP2	1	If my distance to coordinates (200,100,100) is greater than 5 and my state is 'cap1', fly to coordinates (200,100,100).
CAP2	1	If my distance to coordinates (200,100,100) is less than 5 and my state is 'cap1', change my state to 'cap2'.
flyToCAP1	1	If my distance to coordinates (200,-100,100) is greater than 5 and my state is 'cap2', fly to coordinates (200,-100,100).
CAP1	1	If my distance to coordinates (200,-100,100) is less than 5 and my state is 'cap2', change my state to 'cap1'.
engageRWR	2	If my RWR detects an enemy radar, and I have no enemies on my radar, and my radar is in search mode, then turn approximately towards that radar.
engageRadar	2	If I have an enemy on my radar, then turn approximately towards this enemy.
lock	5	If I have an enemy on my radar, and my distance to this enemy is less than 90, set my radar to 'lock' mode and turn towards this enemy.
fire	6	If my radar is in 'lock' mode, and my distance to the enemy is less than 100, and this enemy is alive, and there is no missile flying towards this enemy, and I have missiles left, then fire a missile at this enemy.
support	7	If I have an enemy on my radar, and there is a missile flying at this enemy, set my radar to 'lock' mode and turn towards this enemy.



Table 5. Rule modifications for the alternative versions of the tactics used by 'red'.

<b>Rule name</b>	<b>Pr.</b>	<b>Description</b>
initialCAP1 ( <i>changed</i> )	1	In all cases, change my state to 'cap2', and set my radar to 'search' mode.

Table 6. Rule modifications for the Evading tactic used by 'red'.

<b>Rule name</b>	<b>Pr.</b>	<b>Description</b>
evadeMissile180 ( <i>added</i> )	9	If there is a missile flying at me, and my Radar Warning Receiver detects an enemy radar, turn approximately 180 degrees.

Table 7. Rule modifications for the Short Range tactic used by 'red'.

<b>Rule name</b>	<b>Pr.</b>	<b>Description</b>
fire ( <i>changed</i> )	6	If my radar is in 'lock' mode, and my distance to the enemy is less than 50, and this enemy is alive, and there is no missile flying towards this enemy, and I have missiles left, then fire a missile at this enemy.