

Centralized Versus Decentralized Team Coordination Using Dynamic Scripting

Toubman, Armon; Roessingh, Jan Joris; Spronck, P.H.M.; Plaat, Aske; van den Herik, H.J.

Published in:

Proceedings of the 28th European Simulation and Modelling Conference, ESM'2014

Document version:

Peer reviewed version

Publication date:

2014

[Link to publication](#)

Citation for published version (APA):

Toubman, A., Roessingh, J. J., Spronck, P. H. M., Plaat, A., & van den Herik, H. J. (2014). Centralized Versus Decentralized Team Coordination Using Dynamic Scripting. In *Proceedings of the 28th European Simulation and Modelling Conference, ESM'2014* EUROSIS.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy

If you believe that this document breaches copyright, please contact us providing details, and we will remove access to the work immediately and investigate your claim.

CENTRALIZED VERSUS DECENTRALIZED TEAM COORDINATION USING DYNAMIC SCRIPTING

Armon Toubman, Jan Joris Roessingh
National Aerospace Laboratory
Anthony Fokkerweg 2
1059 CM Amsterdam, Netherlands
{Armon.Toubman,
Jan.Joris.Roessingh@nlr.nl}

Pieter Spronck
Tilburg University
P.O. Box 90153
5000 LE Tilburg, Netherlands
p.spronck@gmail.com

Aske Plaat, Jaap van den Herik
Leiden University
Niels Bohrweg 1
2333 CA Leiden, Netherlands
{aske.plaat,
jaapvandenherik@gmail.com}

KEYWORDS

Machine Learning, Reinforcement Learning, Air Combat Simulation, Dynamic Scripting, Team Coordination

ABSTRACT

Computer generated forces (CGFs) must display realistic behavior for tactical training simulations to yield an effective training experience. Traditionally, the behavior of CGFs is scripted. However, there are three drawbacks, viz. (1) scripting limits the adaptive behavior of CGFs, (2) creating scripts is difficult and (3) it requires scarce domain expertise. A promising machine learning technique is the *dynamic* scripting of CGF behavior. In simulating air combat scenarios, team behavior is important, both with and without communication. While dynamic scripting has been reported to be effective in creating behavior for single fighters, it has not often been used for team coordination. The dynamic scripting technique is sufficiently flexible to be used for different team coordination methods. In this paper, we report the first results on centralized coordination of dynamically scripted air combat teams, and compare these results to a decentralized approach from earlier work. We find that using the centralized approach leads to higher performance and more efficient learning, although creativity of the solutions seems bounded by the reduced complexity.

INTRODUCTION

Our point of departure is the problem of controlling computer generated forces (CGFs) in military training simulations. These CGFs need to exhibit realistic behavior for simulations to have the highest possible educational value. In the real world, military units often operate in teams, such as infantry fireteams, carrier battle groups, or air force flights. The coordination between team members is of specific interest to our research. In earlier work (Toubman, Roessingh, Spronck, Plaat, & Van den Herik, 2014), we investigated team coordination by using communication between CGFs. The CGFs then automatically learn team behavior, including the required elements of communication, using a machine learning technique called Dynamic Scripting (DS).

DS is a reinforcement learning technique that tries to find optimal combinations of behavior rules in a rule base. Since DS works with predefined behavior rules, the learning process is more transparent than with, e.g., subsymbolic

methods. In (Toubman et al., 2014), we took advantage of this transparency by implementing a communication scheme inside the behavior rules used by DS to study the resulting communication patterns. So, the DS algorithm would automatically discover which messages and responses led to winning behaviors.

Our previous work required setting up to two concurrently learning agents, resulting in two instances of DS of which the learned rules were interdependent. This complicated interpreting the resulting rules and may have complicated rule convergence. To reduce the complexity of the setup, both computationally and for the human maintainer, in the current paper we investigate a setup with a single instance of DS. By using one instance of DS to control the agents, we are essentially transforming the agent system to a centralized system. The main research question of this paper is therefore: “Does using this centralized system actually reduce the complexity, and will it result in a more effective and efficient learning process?” The paper is, as far as we know, the first to study centralized communication of dynamic scripts. We find that, in general, it significantly outperforms previous work.

RELATED WORK

Coordination in multi-agent systems has been a subject of active research since the rise of interest in distributed artificial intelligence (Ossowski & Menezes, 2006). Various models, languages and applications have been developed over the years. So far, the lesson learned is that there is no one-size-fits-all solution.

Coordination methods can be divided into centralized and decentralized methods. Coordination is called centralized when a single actor collects information and decides on the right action for all agents in the system. The choice between centralized and decentralized methods is not a clear one, as it has been shown that sometimes both methods can successfully be applied (Jennings, Sycara, & Wooldridge, 1998; Panait & Luke, 2005).

The difficulty of designing coordination methods has called for the use of machine learning. Through machine learning, agents can learn (1) what actions to coordinate and (2) how to coordinate them. Various methods for learning coordination have been developed. Examples include (Balch & Arkin, 1994; Biggers & Ioerger, 2001; Bonarini & Trianni, 2001; Kidney & Denzinger, 2006). Ho & Kamel

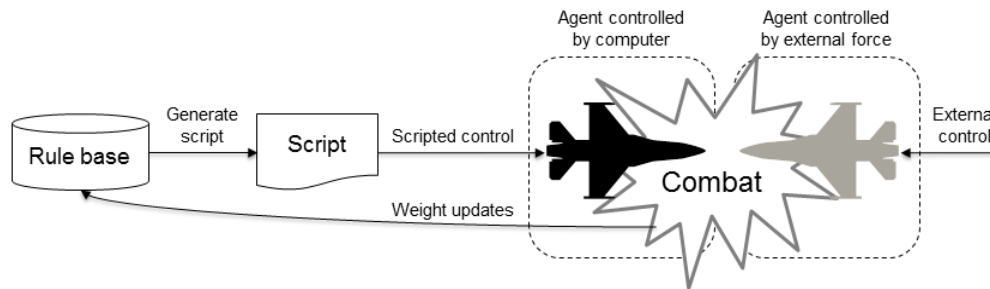


Figure 1: Diagram of the Dynamic Scripting Learning Process

(1998) present a classification of coordination methods, along with common problems: (1) determining convergence, (2) the complexity of the methods, and (3) attaining good performance. Additional problems include (4) dealing with extra agents, and (5) cross-domain applicability.

Most agent coordination research focuses on abstract, puzzle-like domains with a limited number of possible actions, such as the well-known predator-prey domain (Ho & Kamel, 1998; Stone & Veloso, 2000). Instead, we are looking to generate behavior for agents in air combat simulations, with complex environments including hostile agents, and an array of actions with various parameters. Recent efforts in this domain are few. Among others they include the use of neural networks (Su, Lai, Lin, & You, 2012; Teng, Tan, Ong, & Lee, 2012) and differential evolution (Salling, Rensfelt, Stensbäck, & Ögren, 2013). However, these methods lack transparency, which is in our view an essential property of behavior models for training simulators, as they need to be understood by training instructors. This is the main reason why we turned to DS for our research.

DS (Spronck, Ponsen, Sprinkhuizen-Kuyper, & Postma, 2006) is an online learning technique based on reinforcement learning. The learning process is initiated with a rule base that contains behaviour rules for a certain agent. DS selects a certain number of rules from this rule base using weighted random selection. The selected rules form a script that governs the behaviour of an agent during a trial. Then the weights of the rules that were activated during the trial are adjusted based on the agent's performance. This way, the chance to be selected again increases for rules that lead to wanted behaviour, and decreases for rules that lead to unwanted behaviour. Full details on the DS algorithm can be found in (Spronck et al., 2006).

In (Toubman et al., 2014), we extended regular DS with a team coordination method which we called "DS+C." In DS+C, agents are allowed to communicate with each other. The communication is done through behavior rules. In this way, the DS algorithm was able to learn which messages and which responses lead to good behavior. With DS+C, the agents were able to win more encounters than with regular DS, and do so earlier on in the learning process. This was especially the case against an unpredictable opponent. Agents using DS+C were better equipped to handle an opponent that used different tactics throughout the learning process.

DS+C has only been tested on a small scale. The experiments in (Toubman et al., 2014) used a 2-versus-1 scenario in which two learning 'blue' agents intercepted one 'red' agent that used a static script. Both blues had their own rule base together with their own instance of the learning algorithm. In essence, learning took place concurrently, and in a distributed manner (Jennings et al., 1998; Sen & Weiss, 1999). While this scheme produced good results, it is not expected to scale easily, and resulting traces proved difficult to interpret.

While DS is not computationally expensive per se (the actual simulations require the largest share of the runtime), interlinking multiple instances such as in DS+C creates an increasingly difficult optimization problem as both agents try to optimize their rule selections simultaneously. Also, on a design level, it is easy to add a new agent with a self-contained rule base, but it is harder to design a rule base for a learning agent that simultaneously has to learn to coordinate with other agents. This difficulty increases with the number of agents and functional requirements (Turner & Jennings, 2001). So, while multiagent systems are inherently scalable (Stone & Veloso, 2000), the issue here is that DS+C agents require knowledge in their rule bases about the other team members. This is not only a problem when designing rules for the agents before any learning takes place, but also during a possible review phase, when the agents have learned and their behaviour is reviewed and manually tweaked. For these reasons, we tried to reduce the complexity DS+C while keeping its benefits, by moving to centralized coordination.

DYNAMIC SCRIPTING WITH CENTRALIZED TEAM COORDINATION

The overarching goal of this type of research and previous research is to generate behavior for CGFs in training simulations in an easy way using machine learning. To increase the realism of these CGFs, we looked at adding team coordination (Toubman et al., 2014). Therefore, we have added coordination rules to the rule bases of the agents, allowing them to send their own actions and to react to the others intentions.

For the approach presented in this paper, we addressed the scaling issue by reducing the number of learning agents to one per team. The 'master' agent will direct one or more 'slave' agents through the same communication mechanism used in DS+C. However, in the new case, only the master agent will optimize its own rule base, that will govern both

the behavior of the master and the slave agents. In this way, we expect to obtain similar team behavior while reducing the learning complexity.

DS generates scripts by selecting rules from a rule base. In (Toubman et al., 2014) we exploited this mechanism by formulating rules that send out messages in addition to other effects, and thereby enabling DS to find effective exchanges of messages. In the current paper, we use the same mechanism in the same way, only in one direction. We will refer to this new method as the centralized team coordination method (CTCM). It consists of two phases.

First, a rule base is designed for the master agent with behavior rules of which certain combinations might be able to solve partly the task at hand. In the case of our air combat simulation, such rules allow the agent to evade incoming missiles, or make heading changes to avoid detection. Creating several variants of each of those rules enables a wider range of possible behavior. As with DS+C, executing these rules also makes the agent broadcast a message to its slave agents with the intention of its current action.

Second, a separate rule base is designed for the slave agents. Apart from some default rules with fallback behavior, the rules in the rule base are triggered only by the reception of messages from the master agent. For the slave agents, variants of rules can be added, to define a wider range of behavior. These variants have to be triggered by distinct messages from the master agent. The rule base of the master agent can also contain rules that are duplicates of each other, except for the message that they send. This way different behaviors for the slave agent can be tried while keeping the behavior of the master agent the same. The DS rule selection process allows us to try out automatically various combinations of behavior in the master and slave agents.

Of course, to make decisions, agents need to be able to observe their environment. The master agent bases its decisions for its own behavior and for the slaves' behavior on its own observations. In our case this happens through sensors such as the radar. However, if the slave agents are also capable of making observations, it makes no sense to ignore this information. Therefore, the slave agents are allowed to send messages back to the master agent using rules. The master agent does not actively respond to these messages, but uses the information from the slave agents to base new decisions on. This gives the master agent a wider view, but not a completely global view on the environment of its team.

The rule base of the master agent is optimized using DS, while the rule bases of the slaves are not. Throughout the learning process, the DS algorithm selects rules from the rule base and forms a script for the master agent. The master agent uses this script during a trial in its environment. The selected rules also implicitly dictate the behavior of the slave agents.

METHOD

The application of the CTCM was tested in the same custom air-to-air combat simulation used in (Toubman et al., 2014),

together with the same parameters for DS, allowing a close comparison. The approach is summarized there as follows. In the simulation, a formation of two blue fighters ("the blues", i.e., a lead and a wingman) had to eliminate a single red fighter. The red fighter flew a Combat Air Patrol (CAP) (see Figure 2) to defend an area of airspace. The mission of the blues was considered successful if they eliminated the red fighter without any losses on their own side. The mission of red was to eliminate all fighters it detected. Figure 3 shows a screenshot of the simulation.

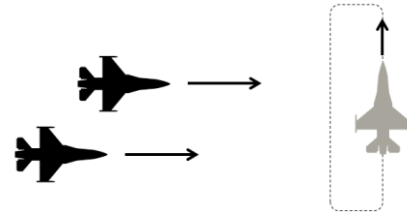


Figure 2: The 'Blues' (left) Fly Towards 'Red' (right), Who Is Flying a CAP

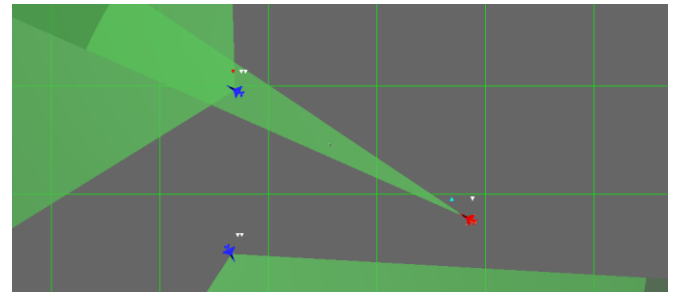


Figure 3. Screenshot of the Simulation

For this application, the blue lead was made the master agent, and the blue wingman was made the slave agent. The lead used two default rules: one rule to let the lead fly to the area where red flies its CAP, and also to send a message to the wingman to let it fly in formation, and the other rule to set the lead's radar to search (wide angle) mode. Both rules were only used when no other rules applied. The default rules were added to each script generated by the DS algorithm. The rule base of the lead further contained rules for locking the radar onto red, rules for firing missiles at red from various distances when the lead had a radar lock on red, and various rules for evading red's missiles and radar locks. Finally, the lead's rule base contained several rules that reacted to messages from the wingman containing certain observations by sending messages for an actual action back to the wingman. All of these non-default rules also send a message to the wingman.

As the wingman did not learn in the case of CTCM, its entire rule base functioned as its script during encounters. The rule base of the wingman mostly contained rules that were activated by certain messages from the lead. For example, these rules included rules for different formations and turning maneuvers. The wingman also had some default rules that did not require messages to activate, such as the following rules (a) to let the wingman fire a missile at red whenever it had the opportunity, or (b) to fly in a default formation with the lead if no other formation message was

Table 1. Turning Points

Tactics of red		TP(50%)			TP(60%)			TP(70%)			TP(80%)		
		μ	σ	p	μ	σ	p	μ	σ	p	μ	σ	p
Mixed	CTCM	42.9	32.7	0.21	65.6	51.9	0.09	100.9	74.9	0.01 *	137.4	82.4	0.00 *
	DS+C	47.9	22.2		77.1	42.5		129.3	71.9		190.6	72.2	
Default	CTCM	21.8	4.2	0.00 *	26.6	12.4	0.00 *	36.9	33.8	0.00 *	60.2	66	0.00 *
	DS+C	31.9	12.2		43	22.7		57.5	34.9		86.4	59.2	
Evading	CTCM	75.7	55.8	0.00 *	87.8	61.9	0.00 *	99.9	66.1	0.47	121.1	72.8	0.12
	DS+C	51	28.2		64.9	31.8		93.9	50		137.2	72.7	
Short Range	CTCM	30.1	19	0.00 *	50	48.9	0.00 *	95.6	87.9	0.00 *	130.9	102.8	0.00 *
	DS+C	42.6	21.3		69.1	46.4		131.1	77.4		195.8	73.9	
Default (alt.)	CTCM	24	8.4	0.00 *	35.4	37.2	0.66	63.5	73.1	0.11	101.4	95.1	0.02 *
	DS+C	30	12		37.2	15.9		50.3	36		75.7	58.8	
Evading (alt.)	CTCM	39.6	23.2	0.01 *	47.5	26.5	0.00 *	65.5	45.8	0.00 *	103.1	76.3	0.01 *
	DS+C	47.2	16.4		60.8	23.1		88.2	52.4		128.8	70.6	
Short Range (alt.)	CTCM	30.5	15.6	0.00 *	56.6	50.6	0.02 *	101.5	88.9	0.00 *	143.2	104.6	0.00 *
	DS+C	49.1	27.5		88.3	55		170.5	76.3		218.1	59.8	

received. Finally, the wingman also used some rules to communicate certain observations to the lead.

For a fair comparison, new rules were added to the DS+C rule bases that allowed the agents to fly in several recently added formations. Apart from these additions, the DS+C rule bases are the same as the original ones used in the previous experiments. (The complete CTCM and DS+C rule bases are omitted here for brevity.)

Red used three basic tactics: a *default* tactic, in which it flies a Combat Air Patrol and engages any intruders it detects; a *short range* tactic, which is the same as the *default* tactic but red is only allowed to fire missiles from a close distance; and an *evading* tactic, which is the same as the *default* tactic but red tries to evade incoming missiles. *Alternative* tactics were also added, which were the same as the three basic tactics but in which red flies the CAP in the opposite direction. Finally, red was given a *mixed* tactic which consisted of the three basic tactics and their alternatives. When using the *mixed* tactic, red would use one of the six tactics until it lost an encounter, at which point it would switch to a new randomly selected tactic.

Because the CTCM only employs one learning agent, instead of the two learning agents used in DS+C, we conjecture that CTCM will be able to learn faster than DS+C. However, as DS+C has two agents trying out different combinations of rules, it is expected that DS+C can be more creative than CTCM and therefore come up with better solutions. In other words, we conjecture that the CTCM will be more efficient but not more effective than DS+C.

To measure effectiveness, we can simply look at resulting win/loss ratios. However, it is hard to determine the efficiency of agents using DS, due to the random sampling of rules during the learning process. Therefore, we re-use the TP(X) measure from (Toubman et al., 2014). In brief, we

look at the results with a moving window of 20 consecutive encounters. Once blue reaches X% wins in this window, we say that a certain *turning point* has been reached in the learning process. By varying X we can adjust the strictness of this measure.

To investigate the efficiency and effectiveness of the CTCM, and compare them to that of DS+C, simulations were run in which the blues used either of these methods, and red used one of its seven tactics.

RESULTS

For each of the seven tactics, results were averaged over 100 learning episodes. Each learning episode consisted of 250 encounters.

For each tactic of red, the TP(X) was calculated for the blues, with X being 50%, 60%, 70% and 80%. These values were chosen because they represent the most interesting range; below 50% blue is still losing, and above 80% blue will be experiencing rare winning streaks.

The mean TP(X) results, together with the standard deviations, are shown in Table 1. Two-tailed two-sample *t*-tests were performed on the pairs of methods for each tactic. Table 1 shows the *p* values for pairs that differ significantly at the $\alpha = 0.05$ significance level (indicated with asterisks). In 21 out of the 28 comparisons, a significant difference is found, of which 19 are in favor of CTCM.

Figure 4 shows the cumulative sum of blue wins using both coordination methods, measured over all tactics. The CTCM reaches 115917 wins out of 175000 total encounters, while DS+C only manages to win 95815.

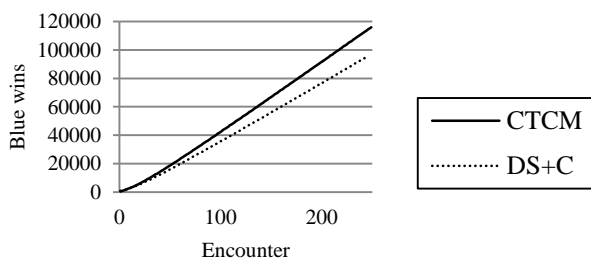


Figure 1. Cumulative Wins

Figure 5 shows rolling averages (window size 20) of blue's win/loss ratio against each of red's tactics individually. The CTCM is able to reach a higher win/loss ratio than DS+C against each tactic, apart from the alternative default tactic. Also, CTCM seems to reach its plateau phase in the learning curves before or at the same time DS+C does. Of special note is the graph for the evading tactic, which does not contain a plateau phase but instead depicts a continuously increasing win/loss ratio.

DISCUSSION

The CTCM was developed as a team coordination method based on DS+C, but with a reduced complexity of learning, which should result in faster and possibly more effective learning. Indeed, it is clear that this is the case. The data in Table 1 shows that learning milestones, measured as turning points, are reached significantly earlier. Figure 4 also supports this notion, showing a steep increase in the number of wins.

Regarding the effectiveness of the coordination methods, Figure 5 shows a distinct difference in performance for five out of seven tactics. For the default (alt.) and mixed tactics, the learning curves end at a very similar performance level.

Before trying to explain these differences, it should be noted that in these experiments, the performance of DS+C does not match that of the original paper (Toubman et al., 2014). After investigation, it was found to be the case that continued improvements of the simulation environment have lead to a slight advantage for the red agent. While this change in performance is disadvantageous for our line of reasoning, the results listed here provide a fairer view.

We conjectured that the CTCM would manage higher efficiency than DS+C. This also turned out to be the case. Because the CTCM only employs one learning agent, the non-learning agent becomes a predictable factor in the environment of the learning agent. In the case of DS+C, with two learning agents, the learning agents will have to adjust their behaviour to the other's behaviour as well, increasing the complexity of the learning problem. With the CTCM, less combinations of behaviour rules are possible. Assuming two agents, a script size s , and a rule base containing r rules for both agents, at the moment the DS algorithm selects rules to be used, there are $\binom{r}{s}$ possible combinations of scripts for the CTCM, whereas for DS+C, this number increases to $\binom{r}{s}^2$. Without taking into account the weight optimization

performed by DS, this already indicates that optimal scripts are likely to be found faster when using CTCM.

The smaller amount of possible scripts should also result in less creativity in the development of solutions against the enemy's tactic. Therefore, we conjectured that CTCM would not be more effective than DS+C. However, the opposite is the case, as CTCM showed higher win/loss ratios, and a higher total amount of wins. A possible explanation is again the lower number of combinations of scripts. As CTCM rapidly proceeds to find an optimal solution, DS+C struggles to do the same. However, it is interesting to see that against each tactic, CTCM hits a specific cap on performance (0.7 for five different tactics, see Figure 2). This means that even though the performance of CTCM rises rapidly, it is unable to find a perfect solution. This again might be a result of the lower possible creativity. Using the same line of reasoning, it can be argued that the relatively poor performance of DS+C is caused by a too large amount of possible script combinations for the particular problems we are trying to solve (i.e., the seven tactics). If this is the case, there might also be a sweet spot between the rules used with CTCM and DS+C, with a more diverse rule base for more creative solutions, possibly together with a learning method for the second agent as a middle ground between no learning and full DS that does not cause the team behaviour search space to explode.

Finally, there is always the possibility of the quality of the hand-made rules holding back the performance of the agents, but the combination of the DS learning algorithm with ample variations on rules should minimize this effect.

CONCLUSION

In this paper, we presented a centralized team coordination method, that is based on an exchange of messages within a rule-based system such as used in earlier work, but with only one learning agent. The main research question of this paper was: "Does using this centralized system actually reduce the complexity, and will it result in a more effective and efficient learning process?" Based on the results from our experiments, we may conclude that the CTCM is a good alternative to more complex solutions with more learning agents, to let a team of virtual fighter pilots learn effective behavior. The CTCM was able to reach milestones significantly faster, while maintaining a better performance during the learning process. The fact that some of the tactics used by the enemy fighter pilot still present a problem to our learning agents, as evidenced by the difficulty of all agents to reach perfect performance, leaves room for further research.

REFERENCES

- Balch, T., & Arkin, R. C. (1994). Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1), 27–52. doi:10.1007/BF00735341
- Biggers, K. E., & Ioerger, T. R. (2001). Automatic generation of communication and teamwork within multi-agent teams. *Applied Artificial Intelligence*, 15(10), 875–916. doi:10.1080/088395101753242679
- Bonarini, A., & Trianni, V. (2001). Learning fuzzy classifier systems for multi-agent coordination. *Information Sciences*, 136(1-4), 215–239. doi:10.1016/S0020-0255(01)00149-9
- Ho, F., & Kamel, M. (1998). Learning Coordination Strategies for Cooperative Multiagent Systems. *Machine Learning*, 33(2-3), 155–177. doi:10.1023/A:1007562506751
- Jennings, N. R., Sycara, K., & Wooldridge, M. (1998). A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, 1(1), 7–38. doi:10.1023/A:1010090405266
- Kidney, J., & Denzinger, J. (2006). Testing the limits of emergent behavior in MAS using learning of cooperative behavior. In *Proceedings of the 2006 conference on ECAI 2006: 17th European Conference on Artificial Intelligence* (Vol. 141, pp. 260–264). Riva del Garda, Italy: IOS Press.
- Ossowski, S., & Menezes, R. (2006). On coordination and its significance to distributed and multi-agent systems. *Concurrency and Computation: Practice and Experience*, 18(4), 359–370. doi:10.1002/cpe.943
- Panait, L., & Luke, S. (2005). Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multi-Agent Systems*, 11(3), 387–434. doi:10.1007/s10458-005-2631-2
- Salling, E., Rensfelt, A., Stensbäck, N., & Ögren, P. (2013). Learning Air Combat Parameters using Differential Evolution. In *Twelfth Scandinavian Conference on Artificial Intelligence* (pp. 225–234). doi:10.3233/978-1-61499-330-8-225
- Sen, S., & Weiss, G. (1999). Learning in Multiagent Systems. In G. Weiss (Ed.), *Multiagent Systems* (pp. 259–298). MIT Press.
- 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.
- Su, M.-C., Lai, S.-C., Lin, S.-C., & You, L.-F. (2012). A new approach to multi-aircraft air combat assignments. *Swarm and Evolutionary Computation*, 6, 39–46. Retrieved from <http://www.sciencedirect.com/science/article/pii/S2210650212000260>
- Teng, T.-H., Tan, A.-H., Ong, W.-S., & Lee, K.-L. (2012). Adaptive CGF for pilots training in air combat simulation. In *15th International Conference on Information Fusion (FUSION)* (pp. 2263–2270). Singapore.
- Toubman, A., Roessingh, J. J., Spronck, P., Plaat, A., & Van den Herik, J. (2014). Dynamic Scripting with Team Coordination in Air Combat Simulation. In *Proceedings of the 27th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems* (pp. 440–449). Kaohsiung, Taiwan: Springer-Verlag. doi:10.1007/978-3-319-07455-9_46
- Turner, P. J., & Jennings, N. R. (2001). Improving the scalability of multi-agent systems. In T. Wagner & O. F. Rana (Eds.), *Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems* (pp. 246–262). Springer Berlin Heidelberg. doi:10.1007/3-540-47772-1_25

ARMON TOUBMAN is a PhD student at the Training, Simulation and Operator Performance department of the National Aerospace Laboratory in the Netherlands. His research focuses on the use of machine learning techniques for the automatic generation of behavior for air combat simulations. He holds a Master of Science degree in Artificial Intelligence from VU University Amsterdam.

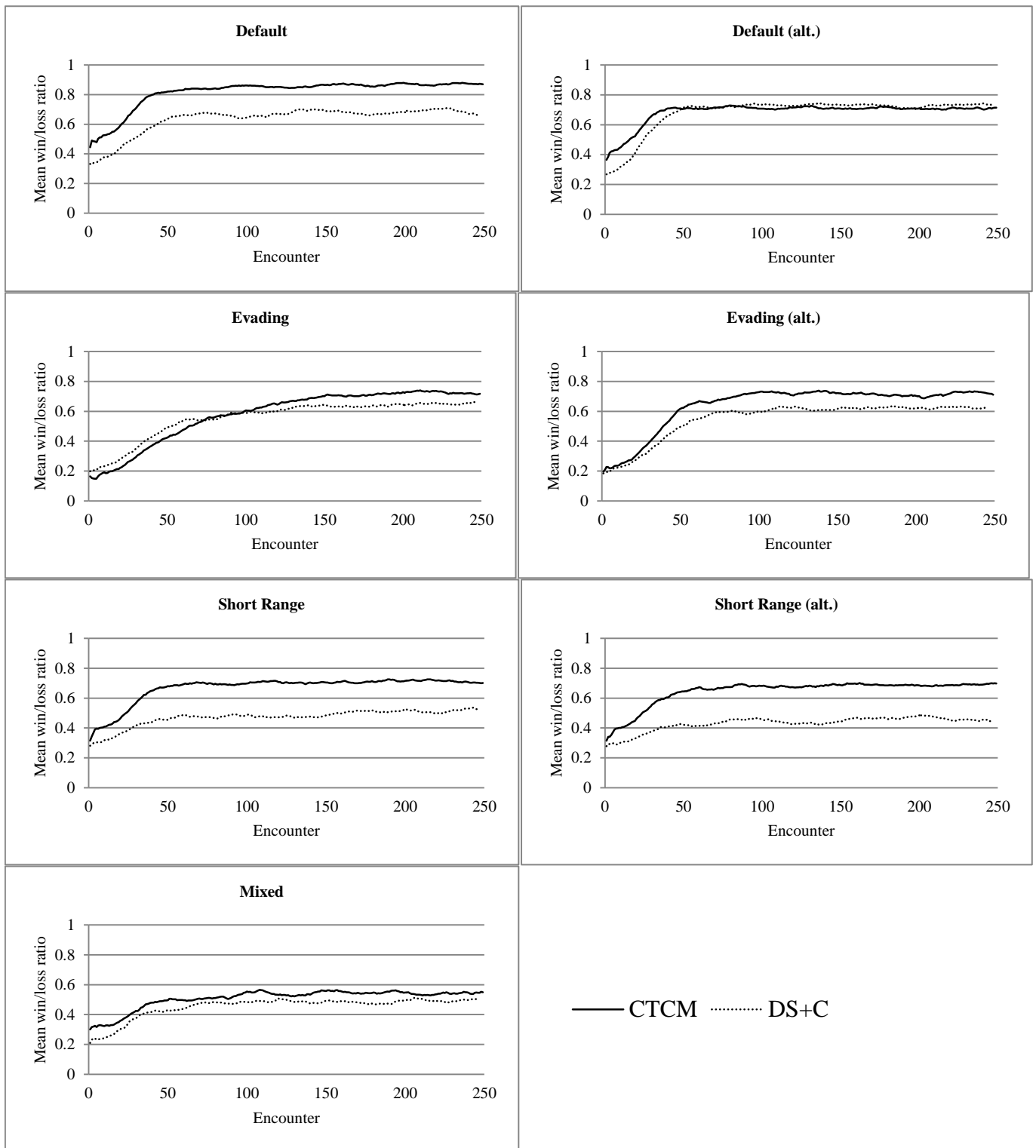


Figure 2. Learning Curves