Calculated Moves: Generating Air Combat Behaviour

Toubman, A.

3 Team coordination

In this chapter we investigate research question 1: *To what extent can we generate air combat behaviour models that produce team coordination?*

Team coordination is an essential part of air combat. We will consider what team coordination means, by reviewing it from two perspectives: (1) the air combat perspective, and (2) the multi-agent system perspective. The field of multi-agent systems will provide us with a framework by which we can implement a variety of coordination methods. Subsequently we will experiment with them in our air combat simulations.

This chapter is structured as follows. First, we elaborate on the concept of team coordination from the two perspectives mentioned above (Section 3.1). Based on these perspectives, we develop and present three coordination methods: (1) TACIT, (2) CENT, and (3) DECENT. Then, we implement the three methods in a dynamic scripting environment (Section 3.2). Next, we determine the effect of the three coordination methods on the performance of a pair of CGFs by means of an experiment involving automated simulations (Section 3.3). We present the results of the experiment (Section 3.4) and then discuss them (Section 3.5). Finally, we summarise the chapter and answer research question 1 (Section 3.6).

This chapter is based on the following two publications.

- A. Toubman, J. J. Roessingh, P. Spronck, A. Plaat and H. J. Van den Herik (2014a). Dynamic Scripting with Team Coordination in Air Combat Simulation. In: *Modern Advances in Applied Intelligence: 27th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2014, Kaohsiung, Taiwan, June 3-6, 2014, Proceedings, Part I*. Ed. by M. Ali, J.-S. Pan, S.-M. Chen and M.-F. Horng. Vol. 8481. Lecture Notes in Computer Science. Kaohsiung, Taiwan: Springer International Publishing, pp. 440–449. ISBN: 978-3-319-07455-9. DOI: 10.1007/978-3-319-07455-9_46
- A. Toubman, J. J. Roessingh, P. Spronck, A. Plaat and H. J. Van den Herik (2014b). Centralized Versus Decentralized Team Coordination Using Dynamic Scripting. In: *Proceedings of the 28th European Simulation and Modelling Conference - ESM'2014*. Ed. by A. C. Brito, J. M. R. Tavares and C. Braganca de Oliveira. Porto, Portugal: Eurosis, pp. 129–134

3.1 Two perspectives on team coordination

In this section, we discuss the concept of team coordination. Team coordination is an extremely broad concept that can be viewed from many different perspectives. Therefore, we restrict ourselves to two relevant perspectives: (1) the air combat perspective and (2) the multi-agent system perspective. The air combat perspective shows us the kind of team coordination that we wish to integrate into the behaviour models of air combat CGFs (Subsection 3.1.1), whereas the multi-agent system perspective provides us with a framework to perform the task (Subsection 3.1.2). Combining the two perspectives will result in three coordination methods: (1) TACIT, (2) CENT, and (3) DECENT. We present the three methods in Subsection 3.1.3.

3.1.1 The air combat perspective

Today, the smallest unit that performs air combat missions is the *two-ship section* (also referred to by *two-ship* or *section*[1]). As the name implies, a two-ship consists of two aircraft. The two aircraft, as well as the pilots that fly them, are called the *lead* and the *wingman* (see, e.g., Borck, Karneeb, Alford and Aha, 2015). In general, the lead makes the tactical decisions for the two-ship, and the wingman follows and supports the lead. To succeed in their missions, the lead and the wingman in a two-ship need to carefully coordinate their actions.

To the best of our knowledge, the book *Fighter Tactics* by Shaw (1985) is the most comprehensive publicly available work on team coordination that takes place in a two-ship. However, the book is several decades old. It has missed substantial technological and doctrinal advances that have happened since its publication. Three of these advances are (1) the improvement of missile guidance and propulsion systems, (2) improved airborne radar technology, and (3) the shift from WVR to BVR air combat, made possible by (1) and (2) (cf. Bongers and Torres, 2014). Shaw (1985) largely focuses on WVR air combat, while BVR air combat is only mentioned in passing. However, the concepts presented by Shaw still surface regularly in modern sources, indicating that these concepts are also relevant today (cf. Bigelow, Taylor, Moore and Thomas, 2003; Aleshire, 2005; Crane, Bennett Jr, Borgvall and Waldelöf, 2006; Marken, Taylor, Ausink, Hanser and Anderegg, 2007; Hinman, Jahn and Jinnette, 2009; Laslie, 2015; Stillion, 2015).

According to Shaw (1985), team coordination in two-ships is based on the principle of *mutual support*. Shaw does not explicitly define the principle, but rather illustrates it with examples of behaviour. Here, we abstract Shaw's examples into two concepts: (1) creating situational awareness, and (2) a flexible division of roles. Below, we explain these two concepts.

The first concept, *creating situational awareness*, concerns the gathering of critical information, and using that information to form decisions. Shaw illustrates the importance of creating situational awareness with a straightforward example. A lone fighter aircraft has large blind spots behind and below the aircraft. Therefore, the pilot's ability to visually detect other aircraft

[1] In this thesis, we use the term *two-ship* to avoid confusion with other uses of the term *section*.

is limited. By having fighter aircraft fly in pairs, the blind spots of both aircraft are to a large extent reduced. Although visual detection has nowadays been largely replaced by radar, the same concept still applies. Two radars can be used to monitor a larger piece of airspace than is possible by using only one radar. Furthermore, the use of two radars allows, e.g., the tracking of specific target aircraft by one radar, while at the same time searching for additional aircraft using the second radar is possible. By having available more information that is relevant to their situation, the pilots can make more informed decisions on how to act.

The second concept, *a flexible division of roles*, partly follows from the first concept. The lead and the wingman begin their operations with a division of roles that is decided in advance. For instance, during the operations, when the lead of a two-ship is engaged by an opponent and has to perform defensive manoeuvres, the wingman remains free to observe the situation. As the lead is busy defending, the wingman is now in the better position to make tactical decisions for the whole two-ship. Therefore, rather than only observing, the wingman can (1) take over the tactical leadership of the lead, and (2) attack the opponent that is pursuing the lead. In other words, by properly coordinating, a two-ship is able to focus on offensive and defensive actions at the same time.

3.1.2 The multi-agent system perspective

The field of multi-agent systems is concerned with the operation of multiple actors in a single environment, such as CGFs in a simulation environment (see, e.g., Connors, Miller and Lunday, 2016). Team coordination has long been studied as an important part of multi-agent systems. Compared to a single agent, multi-agent systems have multiple benefits, such as (1) the ability to act in more than one physical location, (2) higher fault tolerance, as agents can take over the tasks of other agents in case of failure, and (3) flexibility, offered by the application of specialised agents that can cooperate in various manners (cf. Stone and Veloso, 2000; Yan, Jouandeau and Cherif, 2013; Ye, Zhang and Vasilakos, 2017). However, profitting substantially from these benefits requires that the agents coordinate their actions in some way.

Throughout the literature, coordination methods are classified along two axes: (1) the degree of centralisation of the coordination method, (2) the communication that takes place between the agents. Below, we discuss the two axes.

The first axis is the degree of centralisation of the coordination method. On one extreme of this axis are the methods in which the coordination of the agents is managed in a single, central agent. These methods are commonly called *centralised* coordination methods. In a multi-agent system with centralised coordination, the central agent receives information of all the other agents. It then uses this information to create a global “picture” of the environment. Based on this picture, the central agent plans (what it believes to be) the most optimal actions for each agent. The optimal actions are relayed to the other agents so that the actions can be executed (cf. McLennan, Molloy, Whittaker and Handmer, 2016).

On the other extreme of the same axis we see the methods in which the coordination is managed in a distributed fashion amongst the agents. We refer to these methods as *decentralised* coordination methods. Obviously, decentralised methods do not use a central agent to manage the coordination between agents. Rather, all of the agents in the system coordinate their actions amongst themselves. Often, only a local form of coordination is admitted or required (e.g., between agents working in the same room, or on the same task), as the actions of agents do not interfere with all other agents in the system (cf. Su, Zhang and Bai, 2016; Hou, Wei, Li, Huang and Ashley, 2017b).

The second axis is the communication between the agents. On one extreme of the axis we see agents that do not communicate at all. On the other extreme however, we do not see any well-defined actions either. The reason is that any communication that takes place between agents requires a communication scheme. Since all communication schemes involve a combination of (1) information that is communicated, and (2) a means of communication, a wide range of communication schemes is possible. For example, in terms of information, an agent might be able to communicate its observations to another agent, but not its intentions. At the same time, the available means could be noisy, time-lagged, or only allow a limited number of bytes to be transmitted. Coordination methods and agents that employ communication therefore have to be designed with the communication possibilities taken into account. Jennings, Sycara and Wooldridge (1998, p. 18) summarised the issue of designing communication schemes in a single question: "what and when to communicate?"

As has been implied in this section, there are multiple ways to design coordination methods. Designing a coordination method is one of the most challenging parts of inventing multi-agent systems (cf. Rodriguez-Aguilar, Sierra, Arcos, López-Sánchez and Rodriguez, 2015; Evertsz, Thangarajah and Papasimeon, 2017). The introduction of machine learning into the multi-agent system enables the designers of the system to take advantage of the creativity of machine learning in the design of the coordination method (cf. Tuyls and Weiss, 2012). Machine learning allows agents to develop a well-designed coordination method, and try to improve their operations while using that coordination method as a foundation (cf. Panait and Luke, 2005; Foerster, Assael, De Freitas and Whiteson, 2016; Havrylov and Titov, 2017). In the next section, we combine the two perspectives on team coordination into three coordination methods for CGFs.

3.1.3 Combining the perspectives into coordination methods

In this section, we combine (1) the air combat perspective on team coordination, with (2) the multi-agent system perspective on team coordination. Below, we first explain how we combine the two perspectives. Next, we present three coordination methods that follow from the combination.

From the air combat perspective, we derive the context of the agents for whom we are developing the coordination methods. The agents are (1) a lead CGF and (2) a wingman CGF, that together form a two-ship. Any coordination method for this two-ship should (1) help the

CGFs build situational awareness, and (2) enable a flexible division of roles.

From the multi-agent system perspective, we adopt the two axes along which coordination methods are classified: (1) the extent of the centralisation of the coordination among the agents, and (2) the extent of the communication among the agents. Crossing these two axes results in an axial system that is divided into four quadrants (see Figure 3.1). The four quadrants are:

1. The upper left quadrant, containing decentralised coordination methods without communication.
2. The lower left quadrant, containing centralised coordination methods without communication.
3. The lower right quadrant, containing centralised coordination methods with communication.
4. The upper right quadrant, containing decentralised coordination methods with communication.

Within these quadrants we are able to define particular coordination methods. We define three coordination methods: (1) a decentralised coordination method without communication called TACIT, (2) a centralised coordination method with communication called CENT, and (3) a decentralised coordination method with communication called DECENT. The lower left quadrant is left open without a coordination method. This quadrant requires a centralised coordination method without communication. However, without any form of communication taking place before or during encounters, it is quite difficult to envision what such a centralised coordination method would look like. We consider the development of a centralised coordination method without communication outside of the scope of our research. Below, we describe TACIT, CENT, and DECENT.

TACIT. The coordination method TACIT (see Figure 3.1, upper left) has the following two properties: (1) it is a decentralised method, viz. there is no central agent who controls all actions, and (2) it is a method without communication among the agents. This means that the lead and the wingman each select their own actions in an individualistic manner. They do not purposefully exchange information with each other (hence the term *tacit*). However, the lead and the wingman are able to observe (1) each other, and (2) the results of each other's actions, and then base the selection of their actions on these observations.

The use of TACIT provides an advantage to the designer of the behaviour of the lead and the wingman in the two-ship: (1) the behaviour of the lead can be designed with minimal regard for the wingman, and (2) vice versa (i.e., the behaviour of the wingman can be designed with minimal regard for the lead). However, conversely, the use of TACIT also makes it difficult to include explicitly coordinated interactions in the combined behaviour of the two-ship. Furthermore, it is possible that the behaviour of a team using TACIT will

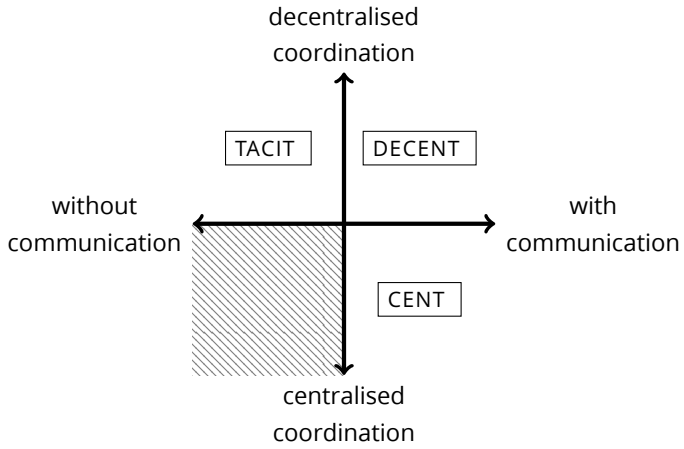


Figure 3.1 The two axes of team coordination: (1) the extent of the centralisation of the coordination (vertical axis), and (2) the extent of the communication between agents (horizontal axis). Placed on the axes are three coordination methods: (1) TACIT, the decentralised coordination method without communication (upper left), (2) CENT, the centralised coordination method with communication (lower right), and (3) DECENT, the decentralised coordination method with communication (upper right). We consider centralised coordination without communication (lower left) to be outside the scope of our research.

resemble least (out of the three coordination methods) the behaviour of a real-world two-ship, as both the lead and the wingman draw their own plan in the simulation.

CENT. The coordination method CENT (see Figure 3.1, lower right) has the following two properties: (1) it is a centralised method, viz. there is a central agent who coordinates all actions, and (2) it is a method with communication among the agents. We designate the lead as the central agent of the two-ship. This means that the lead selects all actions for both itself and the wingman. The actions selected by the lead are sent to the wingman to execute. The wingman helps the lead select actions by sending its observations to the lead. In other words, there is two-way communication between the lead and the wingman.

In CENT, the lead and the wingman are *tightly coupled*, as the lead needs to know the capabilities of the wingman in order to select actions for it. This presents a challenge to the designer of the behaviour of the two-ship. Furthermore, this challenge may grow when, in the future, a designer desires to reuse an existing lead CGF and wingman CGF to form a new two-ship using CENT. If the communication that occurs between the lead and the wingman is incompatible (viz. the lead does not know how to use the observations of the wingman to select actions for the it), the coordination will fail.

DECENT. The coordination method DECENT (see Figure 3.1, upper right) has the following two properties: (1) it is a decentralised method, and (2) it is a method with communication among the agents. By communicating, both the lead and the wingman receive information from each other's sensors, in addition to the information from their own sensors. The additional information leads to better situational awareness, by which both the lead and the wingman can make a better informed selection of actions.

In terms of design, DECENT resembles TACIT albeit with the ability of communication. Therefore, a two-ship that uses DECENT can in principle make a better informed decision (i.e., action selection) than a two-ship that uses TACIT. Furthermore, in contrast to CENT, the lead and the wingman perform their own action selection.

An important topic that we did not mention above is the topic of *learning*. By introducing machine learning to the coordination methods, the two-ship will be able to optimise their coordination with regards to achieving some goal (e.g., defeating an opponent). Machine learning affects the three coordination methods in the following manners. First, a lead and a wingman using TACIT will be able to learn how to act in each other's presence in order to achieve their common goal. Second, in the case of CENT, the central agent will be able to learn (1) how both (the agent itself and the other agent) should act, and (2) what information the central agent has to send to the other agent in order to start these actions. Third, when using DECENT, the lead and the wingman both have to learn individually (1) how to act, and (2) which information to send to each other in order to influence the other agent to act in a more desirable manner.

Returning to the air combat perspective on team coordination, we note that the three coordination methods and the introduction of machine learning now provide both (1) improved situational awareness, and (2) a flexible division of roles with the two-ship. By communicating information to each other (i.e., in the case of CENT and DECENT), the lead and the wingman can help each other build situational awareness. Furthermore, the introduction of machine learning into the coordination methods allows the lead and the wingman to learn how to (1) regulate and act themselves, and (2) coordinate with each other. Because the lead and the wingman share a common goal, we may assume that each of them will in a natural way learn to assume a particular role in the two-ship, such that each will (1) create tactical advantages for both themselves and the other, and (2) take advantage of the created tactical advantages in order to reach the common goal. In essence, the flexible division of roles is caused by the ability to learn.

3.2 Team coordination in dynamic scripting

In this section, we implement the three coordination methods (TACIT, CENT, and DECENT) in a dynamic scripting environment. This entails fitting the concepts of centralisation and communication into the rule-based framework as required by dynamic scripting. By means of the implemented coordination methods, a two-ship of CGFs will be able to learn how to coordinate

their actions with each other. Additionally, for the two methods with communication (CENT and DECENT), the capacity to learn includes the ability to learn what and when to communicate.

We base the implementations in this section on the assumption that the coordinating two-ship will act as the red CGFs in the scenarios that are outlined in Appendix A.4. This means that the goal of the two-ship is to defeat one blue CGF (viz. hit blue with a missile), without being defeated themselves. Below, we describe the implementations of TACIT (Subsection 3.2.1), CENT (Subsection 3.2.2), and DECENT (Subsection 3.2.3) in detail.

3.2.1 Implementing TACIT

In order to implement TACIT, we translate its two main properties to a dynamic scripting environment. These properties are: (1) it is a decentralised method, and (2) it is a method without communication among the agents.

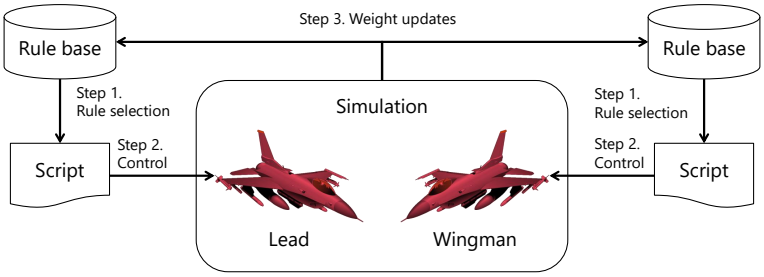
We translate the first property by letting the lead and the wingman in the two-ship be individual learners who each use their own rulebase. We treat the lead and the wingman as equals, in the sense that both receive the same rules in their rulebase (except for the wingman who has additional rules for formation flying, see later in this subsection). We translate the second property by designing the rules of each CGF in such a way that a rule only fires based on the observations made by the CGF to which the rule belongs (viz. no observations are communicated between the CGFs).

Figure 3.2a shows the learning process of a two-ship using TACIT. The lead (left) and the wingman (right) each have their own rulebase. The dynamic scripting algorithm generates a script from each of these rulebases (see Subsection 2.3.3, steps 1 to 3). The scripts are used to control the behaviour of the CGFs in the simulation. Based on the outcome of the simulation, dynamic scripting calculates the weight updates for the rules in the rulebases. During the simulations, there is no communication between the lead and the wingman (in contrast to e.g., DECENT, see Figure 3.2c). Below, we briefly describe the rulebases used by the lead and the wingman.

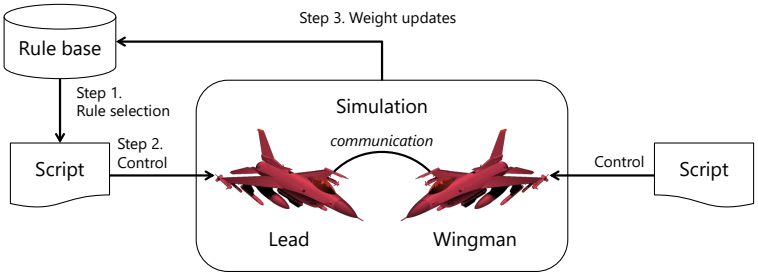
The two rulebases (one for the lead, one for the wingman) are presented in Appendix C. The rules in these rulebases are written in the LWACS scripting language that is described in Appendix B.1. Rather than discussing each individual rule, we divide the rules in each rulebase into three groups: (1) regular rules, (2) filler rules, and (3) default rules. Below, we describe the three groups of rules. At the end of the subsection, we briefly discuss the number of rules that are included in scripts when TACIT is used.

Regular rules. Regular rules are basic behaviour rules that map observations to actions. The lead has regular rules for four distinct types of behaviour: (1) firing missiles (five rules), (2) supporting[2] fired missiles (three rules), (3) evading incoming missiles (three rules), and

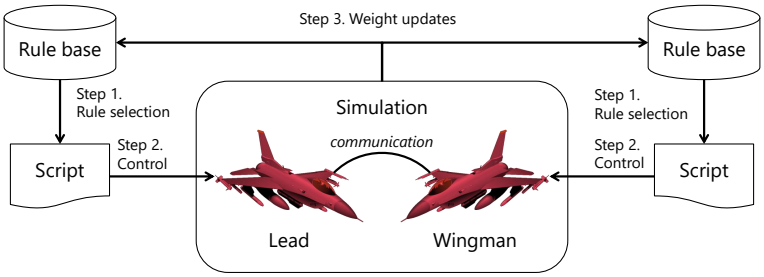
[2]Certain types of real-world missiles have to be *supported*, i.e., guided to the target by the radar of the aircraft that fired the missile. While the missiles in LWACS do not require being supported, we have included rules for this behaviour. The reason is that the behaviour (i.e., tracking the target with a radar and following its movements) may provide a



(a) TACIT. Both the lead and the wingman learn by means of dynamic scripting. There is no communication between them.



(b) CENT. Only the lead learns by means of dynamic scripting. The behaviour of the wingman is controlled by means of a predefined, non-learning script. There is two-way communication between the lead and the wingman.



(c) DECENT. Both the lead and the wingman learn by means of dynamic scripting. There is two-way communication between the lead and the wingman.

Figure 3.2 The three coordination methods implemented in dynamic scripting: (a) TACIT, (b) CENT, and (c) DECENT.

(4) evading opponents that have been detected using the radar warning receiver (RWR) (four rules). The wingman's regular rules include the same rules as the lead, plus rules for a fifth type of behaviour: flying in formation with the lead (five rules). For each type of behaviour, three to five variant rules are included, i.e., rules with slightly different values (e.g., firing from 50, 60, 70, 80, or 90 km).

Filler rules. Filler rules are rules that by design cannot fire, and therefore never execute an action. The purpose of the filler rules is to give the dynamic scripting algorithm the option to fill a script with rules, without forcing the algorithm to include rules in the script for the sake of reaching the required amount of rules. Without filler rules, the algorithm may include rules in scripts that have low weights (viz. they are "bad" rules). Despite their low weights, these rules can trigger during simulations and thereby cause the CGF to perform undesirable actions.

Because the filler rules cannot fire, they are no candidate for a weight increase or decrease when the CGF receives a reward or punishment, respectively. However, the weights of the filler rules are able to change by means of the weight redistribution performed by the dynamic scripting algorithm (see Subsection 2.3.3). In short, dynamic scripting keeps the sum of the weight values in the rulebase constant, by redistributing the weights of all rules whenever a change has to be made in the weight of any rule.

We include 11 filler rules in the rulebases of both the lead and the wingman. The first six filler rules are instrumental for dynamic scripting to fill an entire script with filler rules (because of the script size of six rules, see below). We add five additional filler rules to make the rulebases for TACIT and DECENT contain an equal number of rules. The only difference between TACIT and DECENT is the inclusion of rules for communication in the DECENT rulebases (see Figure 3.2c) Therefore, equalising the number of rules in the rulebases allows for a fair comparison of the learning speeds of the CGFs.

Default rules. Default rules are rules that provide fallback behaviour for the CGFs, when no other rules fire. These rules are automatically appended to every script that is generated by dynamic scripting. The purpose of the default rules is to aid the discovery of combinations of rules. For example, consider a generated script that contains a rule for supporting a missile, but does not contain a rule for firing a missile at any opponent. A CGF using this script will not defeat the opponent, since that requires a missile. By automatically appending a rule for firing a missile (even a sub-optimal rule), the CGF is given a chance to (1) defeat the opponent, (2) give a higher weight to the supporting rule, and (3) retry the supporting rule with a better firing rule from its own rulebase.

The rulebases of both the lead and the wingman include five default rules. Four of these five rules are shared by the lead and the wingman.

tactical advantage in some unexpected other manner.

1. a rule for returning the radar to search mode when the radar no longer detects any opponents,
2. a rule for tracking any opponents, when the radar detects them in search mode,
3. a rule for firing a missile when the lead has a firing opportunity within a range of 40 km,
4. a rule for supporting fired missiles,

lead a rule for flying in the general direction of the blue CGF (to ensure that contact is made between the red and blue teams),

wingman a rule for flying in formation with the lead.

The default rules have been assigned lower priority values (see Subsection 2.3.3) than the regular rules that provide the same behaviour, so that when such a regular rule is included in a script, the default rule never fires.

Script sizes using TACIT

Whenever the dynamic scripting algorithm generates a script from the rules in the rulebase, it only includes a preset number of rules in the script. We call this number the script size s. The script size is an important implementation detail, because it directly affects the complexity of possible scripts, and thus also the complexity of the behaviour of the agents that use the scripts. However, there are no guidelines for selecting a correct script size.

We define the script size of the scripts that dynamic scripting generates for CGFs that use TACIT to be $s = 6$. We choose this script size because it allows scripts to include at least one regular rule for each distinct type of behaviour (see above, *Regular rules*).

Using s and the size of a rulebase, we are able to calculate the number of possible scripts S. The rulebase of the lead contains 26 rules, whereas the rulebase of the wingman contains 31 rules. Here, we do not consider the five default rules that are appended to every generated script. Since a rule can only be included once in the same script, we arrive at the following number of possible scripts (1) S_{lead} for the lead, and (2) $S_{wingman}$ for the wingman.

$$S_{lead} = \binom{26}{6} = 230\,230$$

$$S_{wingman} = \binom{31}{6} = 736\,281$$

The total number of combinations of scripts that the two-ship can use in a simulation is therefore $230\,230 * 736\,281 = 169\,513\,974\,630$. This number indicates the size of the combined search space that the dynamic scripting algorithms (one for the lead, one for the wingman) will have to traverse, in order to find the optimal behaviour of the two-ship (with optimal behaviour seen from the perspective of the reward function, see Subsection 3.3.2). Note that the total

number of combinations of scripts does not equal the total number of different behaviours that the two-ship can display in simulations, because of (a) duplicated rules in the rulebases (e.g., the filler rules) and (b) rules that supersede each other (e.g., firing from at most 80 km also implies firing from 50 km if the opportunity presents itself). An exploratory calculation involving (a) six rules that provide the same behaviour and (b) five rules that supersede each other results in 41,719 possible behaviours for the lead and 192,160 possible behaviours for the wingman. Using these numbers, the total number of possible behaviours for the two-ship is $41\,719 * 192\,160 = 8\,016\,723\,040$.

3.2.2 Implementing CENT

In order to implement CENT, we translate its two main properties to a dynamic scripting environment. These properties are: (1) it is a centralised method, and (2) it is a method with communication among the agents.

We translate the first property by designating the lead as the central agent in the two-ship. The lead learns by means of dynamic scripting. Its rulebase contains rules that (1) define behaviour for itself, and (2) coordinate with the wingman. The wingman can be controlled by any non-learning control method. To stay within the rule-based paradigm, we control the wingman by means of a predefined script. We translate the second property by allowing the CGFs to send messages to each other. The act of sending a message is implemented as an action that can be part of the consequence of a rule. This way, messages can be sent conditionally (i.e., only when a rule with such an action fires). Below, we first describe (A) the rulebase of the lead. Next, we describe (B) the script of the wingman. At the end of the subsection, we briefly discuss the number of rules that are included in scripts when CENT is used.

A: The rulebase of the lead

The rulebase of the lead contains rules that define (1) behaviour for the lead and (2) coordination with the wingman. Regarding the behaviour of the lead, the rulebase includes the same three groups of rules that were defined for the TACIT lead: (1) regular rules, (2) filler rules, and (3) default rules. The only difference is the number of filler rules. The CENT lead's rulebase has six filler rules, rather than the 11 filler rules in the rulebase of the TACIT lead (see Subsection 3.2.1).

For the coordination with the wingman, the CENT lead's rulebase contains two additional groups of rules: (4) rules for formation flying (five rules), and (5) directive rules (13 rules). Below, we describe these two groups.

Rules for formation flying. The rulebase of the lead includes five rules for formation flying. These rules are the same as the five regular rules for formation flying which the TACIT wingman has in its rulebase (see Subsection 3.2.1). However, when one of these rules fires, the lead communicates to the wingman that it should assume a certain formation, rather than assuming the formation itself.

Directive rules. Directive rules fire on specific messages that the lead receives from the wingman. We identify the following eight messages that the wingman can send to the lead.

1. The wingman observes a new opponent by means of its radar.
2. The wingman observes a new opponent by means of its RWR.
3. The wingman observes that a missile is flying towards the wingman.
4. The wingman is able to fire a missile at an opponent from <50 km away.
5. The wingman is able to fire a missile at an opponent from <60 km away.
6. The wingman is able to fire a missile at an opponent from <70 km away.
7. The wingman is able to fire a missile at an opponent from <80 km away.
8. The wingman is able to fire a missile at an opponent from <90 km away.

Upon firing, each directive rule causes the lead to send an instruction to the wingman. In response to each of the first three messages, the lead instructs the wingman to perform one of three actions.

1. Turn right ninety degrees,
2. turn left ninety degrees, or
3. turn towards the observed opponent (in case of message 1 or 2).

In response to the remaining five messages, the lead instructs the wingman to fire. Note that the lead does not respond to the messages the same way in each simulation (or at all), since the directive rules are subject to dynamic scripting's rule selection.

B: The script of the wingman

The wingman uses a predefined script to control its behaviour. Consequently, all of the rules in the script are able to fire in each simulation. The script of the wingman consists of three groups of rules: (1) informative rules, (2) executive rules, and (3) default rules. Below, we describe the three groups of rules.

Informative rules. Informative rules send a message to the lead to inform it of a new observation made by the wingman. The script includes eight informative rules. Five of the informative rules inform the lead that the wingman is able to fire at the opponent from within a certain distance (50, 60, 70, 80, or 90 km). The remaining three informative rules inform the lead of one of three events.

1. The wingman has detected a new opponent by means of its radar,
2. the wingman has detected a new opponent by means of its RWR, or

3. the wingman has detected that a new missile has been fired at it (see A, *Directive rule*).

Executive rules. Executive rules cause the wingman to execute an action upon the reception of an instruction to do so from the lead. The script includes 14 executive rules. Five of the executive rules are for flying in formation (see A, *Rules for formation flying*). Five executive rules are for firing at the opponent (from 50, 60, 70, 80, and 90 km). Two executive rules are for executing evasive manoeuvres (turning right ninety degrees and turning left ninety degrees, respectively). The remaining two executive rules direct the wingman towards the observed position of the opponent.

Default rules. The script of the wingman includes four default rules. Four of these five rules are shared by the lead and the wingman.

1. A rule for returning the radar to search mode when the radar no longer detects any opponents,
2. a rule for tracking any opponents, when the radar detects them in search mode,
3. a rule for firing a missile when the wingman has a firing opportunity within 40 km, and
4. a rule for flying in the general direction of the blue CGF.

Apart from the default rules, the wingman has no capacity to select and execute actions by itself. All actions of the wingman are selected by the lead, and then communicated to the wingman by means of a directive rule. Because the script of the wingman is predefined and does not change between or during simulations, all instructions that are received lead to the firing of the corresponding executive rule.

Script sizes using CENT

We define the script size of the scripts that dynamic scripting generates for the lead CGF to be $s = 12$. This is a larger script size than the script size we defined for CGFs that use TACIT. The reason for the larger script size is that the scripts that are generated for the lead have to include both (1) rules for the lead's behaviour, and (2) rule for the wingman's behaviour.

The rulebase of the lead contains 39 rules, excluding the default rules. Therefore, we arrive at the following S_{lead}.

$$S_{lead} = \binom{39}{12} = 3\,910\,797\,436$$

Since the script of the wingman is predefined (i.e., $S_{wingman} = 1$), the number of possible scripts for the lead is also the number of possible behaviours that the two-ship can display in simulations (viz. the search space for behaviours). This search space is two orders of magnitude

smaller than the search space for TACIT (see Subsection 3.2.1). This suggests that it should be easier for CENT CGFs to find good solutions (viz. faster learning). However, in CENT, only the weights of the rules in one rulebase will be optimised (i.e., for the lead), compared to the two rulebases in TACIT (i.e., for the lead and the wingman). We will look at the specific effects of the centralised method and the two decentralised methods in the experiment that is presented later (see Section 3.3).

3.2.3 Implementing DECENT

In order to implement DECENT, we translate its two main properties to a dynamic scripting environment. These properties are: (1) it is a decentralised method, and (2) it is a method with communication among the agents.

DECENT shares its first property with TACIT, and its second property with CENT. We therefore turn to TACIT and CENT for translating the two properties. First, as in TACIT, we translate the first property by letting the lead and the wingman in the two-ship be individual learners which each use their own rulebase. Second, as in CENT, we translate the second property by allowing the CGFs to send messages to each other, by means of the rules in the rulebases.

Figure 3.2c shows the learning process of a two-ship using DECENT. The lead (left) and the wingman (right) each have their own rulebase. The dynamic scripting algorithm generates a script from each of these rulebases (see Subsection 2.3.3, steps 1 to 3). The scripts are used to control the behaviour of the CGFs in the simulation. Based on the outcome of the simulation, dynamic scripting calculates the weight updates for the rules in the rulebases. During the simulations, there is communication between the lead and the wingman.

We base the DECENT rulebases on the rulebases that are used by the TACIT lead and wingman. We make two changes to these rulebases. The first change is that we add broadcast actions to the rules in the rulebases. The broadcast actions allow the CGFs to communicate to each other what actions they each are performing. The second change is that we replace five of the filler rules by response rules, i.e., rules that only perform actions upon the reception of specific broadcasts. Below, we describe the two changes. At the end of the subsection, we briefly discuss the number of rules that are included in scripts when DECENT is used.

Broadcast actions. We append a broadcast action to each of the regular rules and the default rules (except for default rule 5 of both the lead and the wingman, see Subsection 3.2.1). The appended action sends the *intention* of the rule that fires to the other CGF in the two-ship. We call the appended action the broadcast action because it indiscriminately sends a message to the other CGF, with no regard for whether (1) the sending CGF or (2) the receiving CGF expects the message to be useful. It is up to the rules in the script of the receiving CGF to take action (or not) upon receiving the message (see below, *Response rules*).

As an example of the appended broadcast action, each rule that makes the CGF fire a missile now also sends the message "firing at opponent" to the other CGF. In total, we identify seven intentions: (1) searching for an opponent, (2) tracking an opponent, (3) firing at an opponent, (4) supporting a fired missile, (5) engaging an opponent detecting by means of RWR, (6) evading an opponent detected by means of RWR, and (7) evading a detected incoming missile. The use of broadcast actions to send their intentions to each other gives the CGFs (1) a coarse way to inspect each other's internal state and (2) the means of adjusting their own actions to those of the other CGF in the two-ship.

Response rules. During the design of the response rules, we noticed that it was a difficult task to produce meaningful rules that respond to each of the different broadcasts. Therefore, we only included five response rules. The five response rules are as follows.

1. If the other red CGF is evading an opponent or a missile from that opponent, head towards the approximate location of the opponent.
2. If the other red CGF is tracking the opponent, firing a missile at the opponent, or supporting a missile, then head towards the location of the opponent.
3. If the other red CGF is evading an opponent, also make an evasive manoeuvre by turning 180 degrees.
4. Equal to response rule (3) but turn ninety degrees right.
5. Equal to response rule (3) but turn ninety degrees left.

Script sizes using DECENT

For DECENT, we use the same script size as for TACIT ($s = 6$). Furthermore, the sizes of the rulebases of the lead and the wingman are also equal to those for TACIT (26 and 31 rules, respectively). As a result, the S_{lead} and $S_{wingman}$ for DECENT are equal to those for TACIT as well.

3.3 Experimental setup

We design an experiment to determine which of the three coordination methods leads to the most effective behaviour (i.e, leads to the highest amount of scenarios won). The experiment consists of automated simulations. In this section, we present the setup of the experiment. The setup is divided into six parts: a description of LWACS (Subsection 3.3.1), the red team (Subsection 3.3.2), the blue team (Subsection 3.3.3), the scenarios that were used (Subsection 3.3.4), the independent and dependent variables (Subsection 3.3.5), and a description of our method of analysis and the criteria for comparison (Subsection 3.3.6).

3.3.1 The Lightweight Air Combat Simulator

The simulations in this experiment are performed in the Lightweight Air Combat Simulator (LWACS) simulation program. This program was developed at the Netherlands Aerospace Centre (NLR) for the goal of straightforwardly evaluating behaviour models. Because it was developed at Netherlands Aerospace Centre (NLR), we had complete access to the source code and were able to modify the program to fulfil our simulation needs. The LWACS program is explained in detail in Appendix A. Below, we provide a brief summary of its functionality.

We set up LWACS to simulate an empty section of airspace. This section of airspace is inhabited by fighter jet CGFs. Each fighter jet CGF carries three kinds of devices: (1) a radar by which it can detect other aircraft, (2) four air-to-air missiles that are used for disabling opponent CGFs, and (3) a RWR by which it can detect radar emissions of other aircraft. The physics model used by LWACS is based on classical mechanics, i.e., the CGFs are able to move through the section of airspace without any notion of aerodynamics. Furthermore, we restricted the movement of the CGFs to the horizontal plane in order to simplify the rules that were required for the behaviour of the CGFs. This way, we were able to get a clear first indication of the suitability for dynamic scripting in the BVR air combat domain.

The section of airspace that is simulated in LWACS contains a particular element of randomness which makes the simulations non-deterministic. This element is the *probability-of-kill* of missiles. When a missile hits its target, the probability of the missile disabling its target is calculated based on the distance the missile has flown from the moment of launch. As a result, no two simulated encounters are the same, even though the CGFs in the encounters may use the same behaviour model. To draw conclusions on the effectiveness of the behaviour of the CGFs, we simulate a large number of encounters in the experiment.

In LWACS, behaviour models can be created by means of a scripting language. We present this language in Appendix B. The language is used to write scripts containing behaviour rules. In each encounter that is simulated in LWACS, a script is assigned to each CGF. During the encounter, the script governs the behaviour of the CGF. The presence of existing code for parsing and executing scripts made it simple to integrate the dynamic scripting algorithm directly into LWACS.

3.3.2 Red team

The *red team* (also referred to as *red*) consists of two fighter jet CGFs, a lead and a wingman. The capabilities of the CGFs are described in Appendix A. The goal of the red team is to learn how to defeat the blue team in three different scenarios (see Subsection 3.3.4). The red team uses each of the three coordination methods implemented in dynamic scripting: (1) TACIT, (2) CENT, and (3) DECENT.

The dynamic scripting algorithm requires a reward function to calculate the adjustment of the weights in rulebases. We design a simple reward function, that we call BIN-REWARD. BIN-REWARD stands for *bin*ary *reward*. The reward function provides a reward signal to each CGF with a value

of 1 if the CGF defeats its opponent, and a value of 0 if a member of its own team (including itself) is defeated. The same reward signal is provided to both the lead and the wingman after a simulation. The lead and the wingman then use the reward signal to update the weights in their own rulebases (see Figure 3.2, *Step 3*). We will delve more deeply into the subject of reward functions in Chapter 4.

3.3.3 Blue team

The *blue team* (also referred to as *blue*) consists of a single fighter jet CGF. The capabilities of the CGF are described in Appendix A.2. The goal of the blue team is to defeat the red team, by hitting one of the reds with a missile. The behaviour of blue CGF is governed by scripts (see Appendix A.3).

The use of a team with a single CGF may surprise the reader, as this chapter emphasises the importance of teamwork in air combat. There are three reasons for including only one CGF in the blue team. First, even if teamwork is important, a lone fighter jet armed with air-to-air missiles is still a dangerous opponent. We stress the danger of this opponent by declaring the blue team the winner of a scenario when the blue team hits only one of the red CGFs. Second, at this point, it is unclear how well the red team will perform when encountering any opposing agents, as red's rulebases (and script) are newly designed. The use of a single blue agent is a starting point for evaluating the performance of red. Third, as stated in this chapter and the previous chapter, it is difficult to manually design the behaviour for one agent, let alone two. As a result, the more agents with manually designed behaviour we add, the less confident we can be that together the agents represent a team capable of providing adequate opposition. Later in the thesis, we introduce blue teams with more than one CGF (see Chapter 5 and Chapter 7).

3.3.4 Scenarios

Simulators need to be configured so that a specific situation or event is displayed in the simulation that is run. This configuration is commonly called the scenario (see, e.g., Durak, Topçu, Siegfried and Oğuztüzün, 2014). We define the term scenario below.

Definition 3.1 (Scenario)**.** A scenario is a simulator configuration that defines (1) the number of CGFs in the simulation, (2) the teams that the CGFs belong to, (3) the initial positions, headings, and speeds of the CGFs, and (4) the termination criteria of the simulation.

In the automated simulations that are performed in LWACS, we use the four two-versus-one scenarios described in Appendix A.4: (1) the basic scenario, (2) the close range scenario, (3) the evasive scenario, and (4) the mixed scenario. These four scenarios are repeatedly presented to red. This way, red is able learn how to behave in each of the four scenarios over a run of encounters.

3.3.5 Independent and dependent variables

The experiment uses two independent variables: (1) the three coordination methods (TACIT, CENT, and DECENT), and (2) the four scenarios (basic, close range, evasive, and mixed). The combination of these independent variables results in a 3×4 fully factorial design with twelve conditions. In each condition, we obtain the win rates of red. The win rates are the dependent variable in the experiment.

3.3.6 Method of analysis

We make use of two measures to analyse the win rates of red: we calculate (1) the final performance, and (2) the turning points. Below, we briefly explain the two measures.

Measure 1: Final performance. The win rates of red are expected to show a learning curve, viz. increasing in the beginning of a run of encounters, then levelling off when the scripts generated for red cannot be optimised further. We are interested in measuring the performance of red after it has levelled off, i.e., after red has learned to defeat blue by means of dynamic scripting. We define the final performance of red as red's win rates averaged over the last fifty encounters in each run. The final performance indicates in which condition the reds find the most effective behaviour against blue.

Measure 2: Turning points. We calculate the turning point for each condition. This measure was first presented by Spronck et al. (2006). A sliding window ($n = 10$) is placed over the encounters in each run. The turning point is the last encounter in the sliding window, when the red team has won more encounters in that window than the blue team. We use the turning point as an indication of the learning speed of red.

Because of the use of a sliding window, the turning point measure experiences both a floor effect and a ceiling effect. The floor value of this measure is the nth encounter, which is the case when the turning point is found in the first n encounters. The ceiling value of the turning point measure is encounter n_{max}, where n_{max} is the number of encounters in the run. Even if no turning point exists in the results of a run, a turning point at encounter n_{max} will be measured. The floor and ceiling effects should be taken into account when interpreting the results of the turning point measure.

We investigate the results from the two measures by means of analyses of variance (ANOVAS). Using the ANOVAS, we determine whether red's final performance and/or turning points differ between the three coordination methods and/or four scenarios.

3.4 Experimental results

In this section, we present the results of the experiment.[3] For each condition, we simulated 50 runs of encounters, with each run consisting of 150 encounters. In total, we simulated 90,000 encounters for the experiment.

Figure 3.3 shows the win rates of red against blue. The win rates are divided over the four scenarios. For each scenario, the win rates of red using each of the three coordination methods are shown. The final performance and the turning points of red are shown in Table 3.1 and Table 3.2, respectively. Two two-way ANOVAs were performed.

First two-way ANOVA (final performance)

The first two-way ANOVA was conducted on the influence of the two independent variables (coordination method, scenario) on the final performance of red. All effects were statistically significant at the $\alpha = .05$ level. The main effect of coordination method yielded an F ratio of $F(2, 588) = 16.353$, $p < .001$, indicating a significant difference between the final performances of red using each of the three coordination methods. The main effect of scenario yielded an F ratio of $F(3, 588) = 52.689$, $p < .001$, indicating a significant difference in the final performance of red in each of the four scenarios. Furthermore, the interaction between the coordination methods and the scenarios was significant, $F(6, 588) = 3.844$, $p < .01$. We performed a post hoc Tukey honest significant difference (HSD) test (see, e.g., Holmes, Moody, Dine and Trueman, 2016) to determine the significant differences in final performance between specific pairs of (a) coordination methods and (b) scenarios. First, the post hoc test revealed that (a) the final performance of the reds using CENT differed significantly from that of the reds using either TACIT or DECENT, $p < .001$. Consequently, the final performance of the reds that used either TACIT or DECENT did not differ significantly. Second, the post hoc test revealed that (b) the final performance differed significantly between all scenarios, $p < .001$, except between the close range and mixed scenarios.

Second two-way ANOVA (turning points)

The second two-way ANOVA was conducted on the influence of the two independent variables (coordination method, scenario) on the turning points. The main effect of coordination method was not found to be statistically significant. However, the main effect of scenario was statistically significant at the $\alpha = .05$ level. The main effect of scenario yielded an F ratio of $F(3, 588) = 21.957$, $p < .001$, indicating a significant difference between the turning points of red in the four scenarios. We performed a post hoc Tukey HSD test to determine the significant differences

[3] The results that are reported in this thesis differ from the results that were reported in the two publications. Since these publications, LWACS received multiple updates. The results in this thesis are from new simulations that were run using the latest version of LWACS.

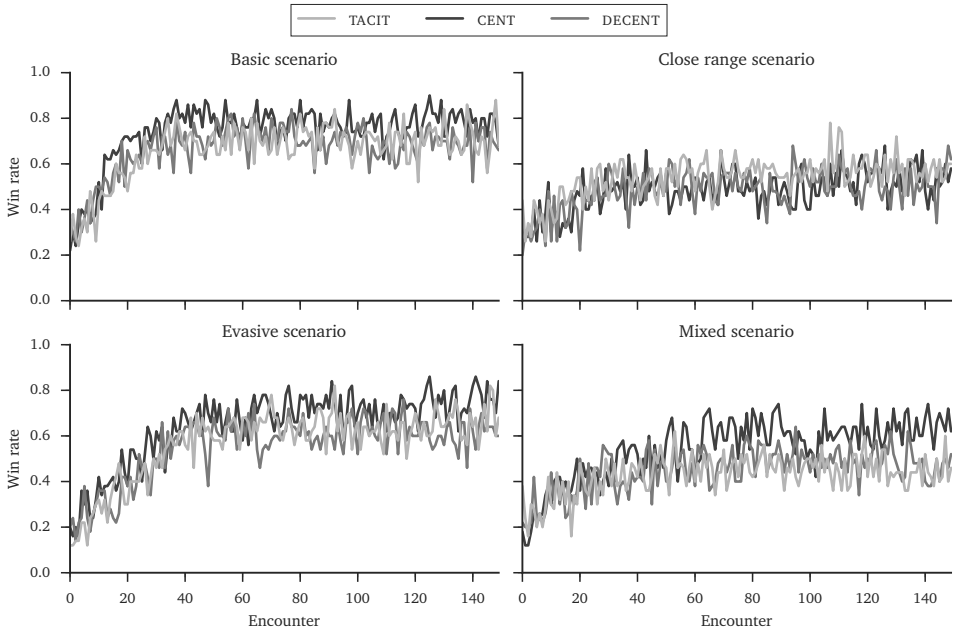


Figure 3.3 The win rates of red against blue. Red used one of three coordination methods: (1) TACIT, (2) CENT, and (3) DECENT. The behaviour of blue depended on four scenarios: (1) the basic scenario, (2) the close range scenario, (3) the evasive scenario, and (4) the mixed scenario.

Table 3.1 The final performance of red. A higher final performance is better.

	Basic scenario		Close range scenario		Evasive scenario		Mixed scenario		Grand mean	
Coordination method	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
TACIT	.704	.213	.584	.187	.643	.190	.452	.117	.596	.202
CENT	.778	.184	.535	.119	.736	.155	.619	.180	.667	.187
DECENT	.686	.161	.521	.149	.621	.157	.486	.117	.579	.166

Table 3.2 The turning points of red. A lower turning point is better.

	Basic scenario		Close range scenario		Evasive scenario		Mixed scenario		Grand mean	
Coordination method	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
TACIT	26.9	13.7	26.9	13.1	36.0	17.4	45.8	30.8	33.9	21.4
CENT	21.4	9.4	33.8	23.9	31.3	20.0	39.8	28.7	31.6	22.5
DECENT	23.2	11.0	40.1	27.4	38.2	15.4	43.5	22.5	36.2	21.4

in turning points between specific pairs of scenarios. The post hoc test revealed that the turning points differed significantly between all scenarios, $p < .01$, except between the close range and evasive scenarios.

3.5 Discussion

In this section, we discuss the results of our experiment. We partition the discussion into four parts: our key finding (Subsection 3.5.1), the effect of centralised coordination on performance (Subsection 3.5.2), the learning process of coordinating CGFs (Subsection 3.5.3), and the way forward (Subsection 3.5.4).

3.5.1 Key finding

Our key finding is that the results suggest that out of the three coordination methods that we tested, the CENT method leads to the most effective behaviour for the red two-ship. In three of the four scenarios (viz. the basic, evasive, and mixed scenarios), the CENT method clearly outperforms both TACIT and DECENT. In these three scenarios, CENT achieves the highest final performance and the lowest turning points, meaning that it is able to find the best rules at the earliest point in the learning process.

3.5.2 The effect of centralised coordination on performance

What sets CENT apart from TACIT and DECENT is that in the CENT method, there is only one CGF (i.e., the lead) that learns by optimising its rulebase. The wingman of this CGF is controlled by means of a non-changing script. In contrast, in both the TACIT and DECENT methods, both the lead and the wingman are each optimising their own rulebases at the same time. Therefore, it appears that the presence of only one rulebase that needs to be optimised is a benefit to the performance of the red two-ship. Since each of the two red CGFs is dependent on the other for winning each encounter (viz. defeating the blue and getting a reward), it takes longer for the CGFs to select rules that complement the rules that are selected by their teammate.

It is only in the close range scenario that the performance of CENT is overtaken by another method. In the close range scenario, it is the TACIT method that reaches both the highest final performance and the lowest turning point. Based on the results, it appears that in this scenario, it is both (a) the decentralisation of the coordination, and (b) the lack of communication that provide an advantage in this scenario. The blue in the close range scenario is scripted to only fire at the reds when it has approached them within a range of 50 km. This leaves the red CGF under fire with little time to effectively evade the missile. Furthermore, while its teammate is being fired upon, the remaining red only has the same amount of time to setup a counterattack while the blue is busy performing its own attack. The data suggests that in tight situations like

this, it is marginally better to forego the communication, and therefore for each red CGF to try to directly take advantage of the situation.

3.5.3 The learning process of coordinating CGFs

To illustrate the learning process of a CGF, we plot the weights of the rules in a rulebase as they change over time. We do so for two rulebases: the rulebase of the red lead using the CENT method in the basic scenario (see Figure 3.4), and the rulebase of the red lead using the CENT method in the mixed scenario (see Figure 3.5). The figures show the progression of the weights of each rule (column) from the first encounter (top row) to the last encounter (bottom row). A darker colour indicates a higher weight. In the figures, the rules are identified by their internal code names for brevity (see the bottom of the figures). The specific rules that are associated with these code names can be found in Appendix C.

We make three observations in Figure 3.4 and Figure 3.5. The first observation is that in both figures, two rules come out with the highest average weights. These rules are (1) `evadeMissile180`, the code name for the rule "if I detect a missile flying at me, turn 180 degrees", and (2) `wingmanAskFireFrom90`, the code name for the rule "if my wingman says it is able to fire from 90 km, tell it to do so". The red lead discovers that these two rules are favourable at an early stage in the learning process.

The second observation is the light patch in Figure 3.4, from code name `fireFrom50` to `support-right`. These rules relate to the firing of missiles. The light patch indicate that the red lead does not favour firing missiles itself, but rather leaves its wingman to fire the missiles to defeat the blue. This is supported by the first observation.

The third observation is that the lead is quite indecisive on the best behaviour for the wingman. This can be seen in both Figure 3.4 and Figure 3.5 in the columns to the right of `wingmanAskFireFrom90`. In both figures, these columns are lightly coloured, meaning that the rules that these columns represent received weight increases in some of the runs over which we averaged, and no weight increases in other runs. While there is seemingly no pattern to the colours in these columns, we can distinguish a small difference. In Figure 3.4, the colours appear smoothed, whereas in Figure 3.5, they appear spotty and irregular. From the smoothed colours we conclude that in different runs of encounters, the rulebase of the lead has converged to different combinations of rules for the behaviour of the wingman. In contrast, the irregular colours show how in the mixed scenario, the lead keeps trying out different combinations of rules for the behaviour of the wingman because the blue opponent continually changes its behaviour. We made similar observations in the rulebases that were used with TACIT and DECENT. As the two figures show, the weights in the rulebases reflect (a) the nature of the scenarios that the rulebases were used in, as well as indicate (b) the flexibility of dynamic scripting, seen from the different combinations of rules that lead to clear divisions in roles between the CGFs.

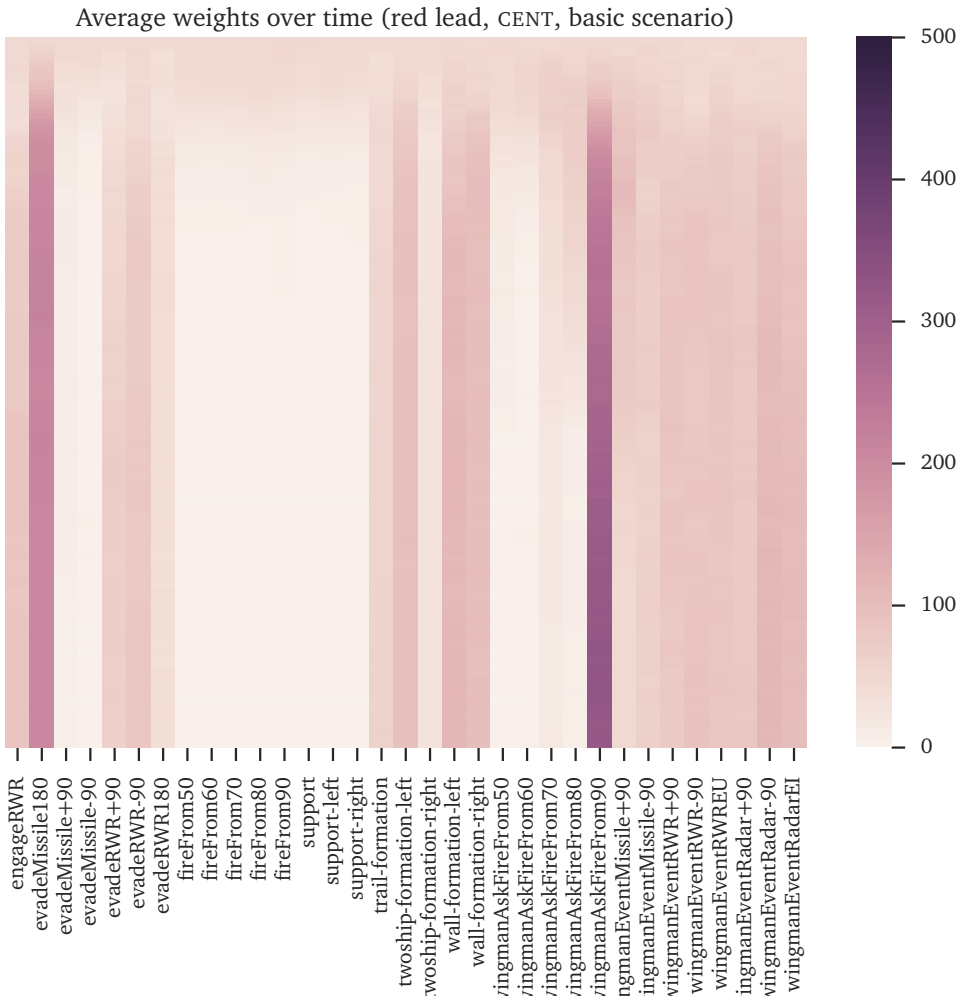


Figure 3.4 The weights of the rules in a rulebase shown as a heatmap graph. The graph shows the progression of the weights (columns) over time (rows, 150 encounters from top to bottom). The colour indicates the value of the weights (see legend on the right). This graph shows the weights in the rulebase of the red lead that used the CENT method in the basic scenario. The weights are averaged over 50 runs of encounters.

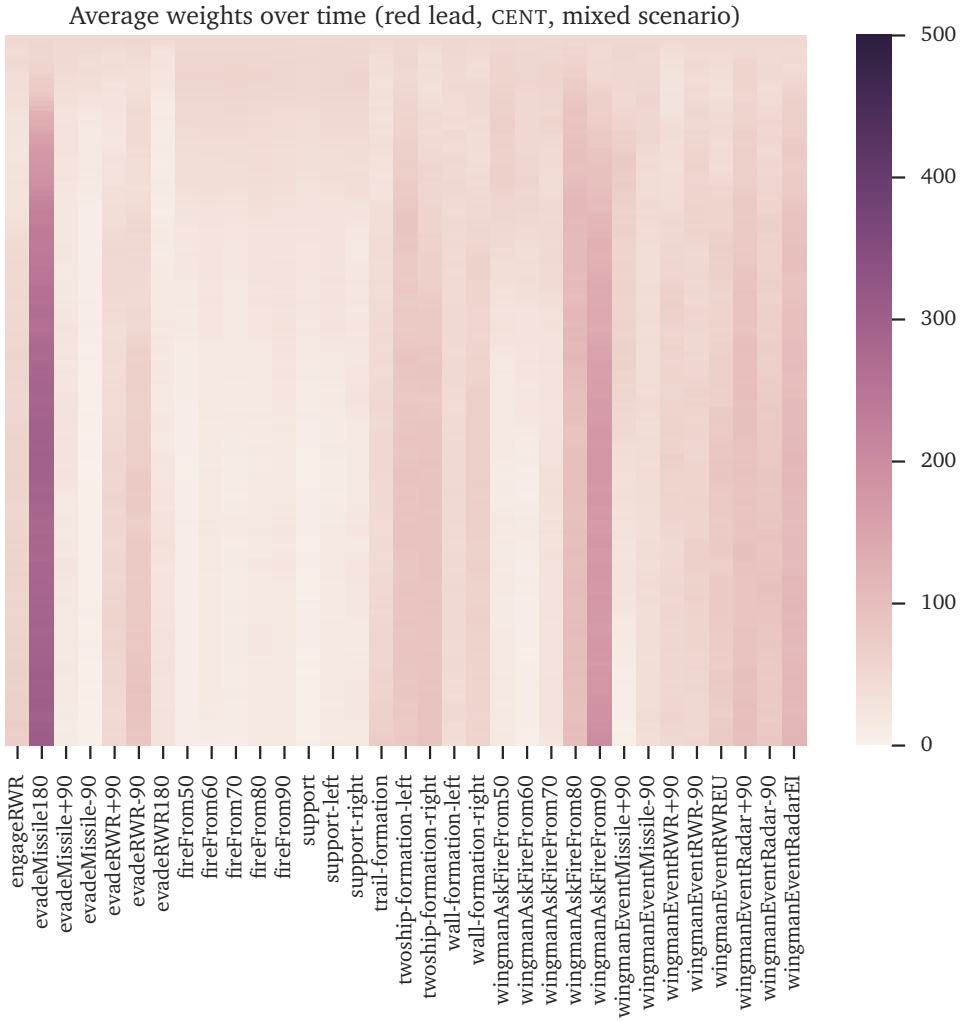


Figure 3.5 The weights of the rules in a rulebase shown as a heatmap graph. The graph shows the progression of the weights (columns) over time (rows, 150 encounters from top to bottom). The colour indicates the value of the weights (see legend on the right). This graph shows the weights in the rulebase of the red lead that used the CENT method in the mixed scenario. The weights are averaged over 50 runs of encounters.

3.5.4 The way forward

Although our results favour the CENT method, we still use a different coordination method in the research that we perform in the remainder of this thesis. The situation is as follows. In Chapters 4 through 7, we use the DECENT method for coordination between the CGFs. We have two reasons for this choice.

First, the use of DECENT is somewhat historical. In early experiments, DECENT provided the best performance. Based on this outcome, the choice was made to use DECENT in our simulations to answer the remaining research questions. However, the outcome presented in this chapter is the result of new simulations (based on the idea that new developments should be tested always as much as possible). Hence, we used the latest version of LWACS, as well as revised rules in the rulebases. Although the outcome surprised us, we decided to continue our research with the methodology chosen (see Section 3.3). Still, we report the current results extensively, since they are candidates for future research in this context.

Second, in our view, DECENT has the most interesting properties out of the three coordination methods presented in this chapter. These properties are (a) the communication between the CGFs, and (b) the fact that both CGFs are trying to optimise their own rulebases at the same time. The two properties open the door for CGFs that (a) have to learn to communicate and cooperate with each other, and (b) each CGF may perhaps use a different machine learning technique. Therefore, out of the three coordination methods, we have taken DECENT as the most attractive method for the research in the remainder of this thesis.

3.6 Answering research question 1

In this chapter, we investigated the generation of behaviour models that allow for team coordination between CGFs. Specifically, we addressed research question 1. Below, we summarise the work done in this chapter, and how this work answers the research question.

Research question 1 reads: *To what extent can we generate air combat behaviour models that produce team coordination?* To answer this question, we reviewed the subject of team coordination from two relevant perspectives: (1) the perspective of air combat, and (2) the perspective of multi-agent systems. Combining these perspectives resulted in three coordination methods (Section 3.1), each of which we implemented in a dynamic scripting environment (Section 3.2): (1) the decentralised coordination method without communication TACIT, (2) the centralised coordination method with communication CENT, and (3) the decentralised coordination method with communication DECENT. The coordination offered by these methods was completely performed within the rule-based framework of dynamic scripting.

In conclusion, we answer research question 1 as follows. The dynamic scripting environment is flexible. It allows us to implement and experiment with three coordination methods: TACIT, CENT, and DECENT. We have shown that each of the three methods enables the CGFs to learn effective

team behaviour in automated simulations. Therefore, the answer to the research question is that we are fully able to generate air combat behaviour models that produce team coordination by means of dynamic scripting. However, out of the three methods, the CENT method clearly provides the best performance. This is apparent in three out of the four scenarios that we used in the automated simulations. Especially in the mixed scenario, the CGFs using the CENT method demonstrate their ability to quickly adapt to the blue's changing behaviour. For this reason, we recommend to first consider the use of centralised coordination in future air combat simulations. Thereafter the scientific comparison between CENT and DECENT can be continued.