Conflict Prioritization with Multi-Agent Deep Reinforcement Learning

D.J.G. Cuppen

Conflict Prioritization with Multi-Agent Deep Reinforcement Learning

by

Daan Cuppen

to obtain the degree of Master of Science
at the Delft University of Technology,

Student number:	4488024	
Thesis committee:	Prof. dr. ir. J.M. Hoekstra,	TU Delft, Supervisor
Thesis committee:	Dr. ir. J. Ellerbroek,	TU Delft, Supervisor
	MSc M.J. Ribeiro,	TU Delft, Supervisor
	Dr. Alessandro Bombelli,	TU Delft, External Examiner

Contents

List of Figures

List of Tables

Nomenclature

List of Abbreviations

ADS-B	Automatic Dependent Surveillance Broadcast
AI	Artificial Intelligence
ANN	Artificial Neural Network
ARV	Allowed Reachable Velocities
ASM	Air Space Management
ATC	Air Traffic Control
ATCO	Air Traffic Controller
ATFM	Air Traffic Flow Management
ATM	Air Traffic Management
ATS	Air Traffic Services
BVLOS	Beyond Visual Line of Sight
CD	Conflict Detection
CDR	Conflict Detection and Resolution
CNN	Convolutional Neural Network
CPA	Closest Point of Approach
CR	Conflict Resolution
DAA	Detect and Avoid
DDPG	Deep Deterministic Policy Gradient
DEP	Domino Effect Parameter
FAA	Federal Aviation Administration
FF	Free Flight
FL	Flight Level
FRV	Forbidden Reachable Velocities
FV	Forbidden Velocities
GS	Ground Speed
ICAO	International Civil Aviation Organisation
IPR	Intrusion Prevention Rate
LoS	Loss of Separation
MARL	Multi-Agent Reinforcement Learning
MDP	Markov Decision Process
MVP	Modified Voltage Potential
NN	Neural Network
ORCA	Optimal Reciprocal Collision Avoidance
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
RV	Reachable Velocities
SSD	Solution Space Diagram
TAS	True Air Speed
TCAS	Traffic Alert and Collision Avoidance System
UAV	Unmanned Aerial Vehicles
UTM	Unmanned Aircraft System Traffic Management
VO	Velocity Obstacles

List of Symbols

α	Step size policy update
ϵ	Exploration Rate
γ	Discount Factor
π	Policy
τ	Look-ahead time window
θ	Model parameters
A	Advantage
a	Action
L	Loss
O	Observation
q	Action-value function
r	Reward
s	State
t	Time
v	Value function

Report Outline

This report contains work from the preliminary and final phase of the MSc Thesis project. The report is divided into two parts:

I **Scientific Article**: The findings of the research in the final part of the thesis are presented in the form of a scientific article.

II **Preliminary Report [already graded]**: In the preliminary phase, a literature study was first conducted. Thereafter, exploratory experiments were conducted to synthesize a research proposal for the final phase of thesis.

I

Scientific Article

Conflict Prioritization with Multi-Agent Deep Reinforcement Learning

D.J.G. Cuppen (MSc Student)
Supervisors: Dr. Ir. J. Ellerbroek, Prof. Dr Ir. J.M. Hoekstra and MSc M.J. Ribeiro
Section Control & Simulation, Department Control and Operations, Faculty Aerospace Engineering
Delft University of Technology, Delft, The Netherlands

***Abstract*—To facilitate an increase in air traffic volume and to allow for more flexibility in the flight paths of aircraft, an abundance of decentralized conflict resolution (CR) algorithms have been developed. The efficiency of such algorithms often deteriorates when employed in high traffic densities. Several methods have tried to prioritize certain conflicts to alleviate part of the problems introduced at high traffic densities. However, manually establishing rules for prioritizing intruders is a difficult task due to the complex traffic patterns that emerge in multi-actor conflicts. Reinforcement Learning (RL) has demonstrated its ability to synthesize strategies while approximating the system dynamics. This research shows how RL can be employed to improve conflict prioritization in multi-actor conflicts. We employ the Proximal Policy Optimization algorithm with an actor-critic network. The RL model decides on intruder selection based on the local observations of an aircraft. It was trained on a limited number of conflict geometries in which it was able to significantly reduce the number of intrusions. A conflict prioritization strategy was then formulated based on the decisions taken by the RL model during training. We show that the efficacy of a conflict resolution algorithm that adopts a global solution, the solution space diagram (SSD) in this research, can be improved when utilizing this conflict prioritization strategy. Finally, these results were compared to the performance of a pairwise CR method, the Modified Voltage Potential (MVP). Even though MVP resulted in a smaller number of intrusions compared to SSD with conflict prioritization, the prioritization strategy did reduce the gap between the two CR methods.**

***Index Terms*—Decentralized Conflict Resolution, Multi-agent Reinforcement Learning, Proximal Policy Optimization, Solution Space Diagram, Modified Voltage Potential, BlueSky ATC Simulator**

I. Introduction

An increment in number of flights is expected in both the manned and unmanned aviation sector [1]–[3]. The workload of the Air Traffic Controller (ATCO) is often seen as the bottleneck that would inhibit a further growth in air traffic volume [4]. One the core tasks of Air Traffic Control (ATC) is Conflict Detection and Resolution (CD&R) [5]. Conflict detection (CD) describes the mechanism which monitors whether conflicts are present. Conflict resolution (CR) is applied to establish a conflict-free trajectory once a conflict is detected [6]. The free flight (FF) concept can facilitate an increase in air traffic volume and allow for a more optimized flight path. The free flight (FF) concept proposes to transition the responsibility for assuring safe traffic separation from ATC to the individual aircraft (airborne separation). CD&R would evolve from a centralized to a decentralized process, decreasing the workload and pressure on ATCOs to guarantee global safety.

A variety of decentralized CD&R algorithms exists. Geometric CR algorithms were found to be very successful in terms of safety and efficiency in a low traffic density. In a conflict with two aircraft, methods such as the Solution Space Diagram (SSD) [7], can achieve a geometrically optimal solution [8]. However, in a high traffic density with multi-aircraft conflicts, the solution can become sub-optimal. The sub-optimal solution is caused by the complex dynamics which arise when multiple aircraft perform conflict avoidance manoeuvres simultaneously. In the case of SSD, all conflicts are considered simultaneously and a global solution is adopted. In a multi-conflict situation, the only solution could be a sharp turn away from the target or in the extreme no feasible solution exist.

Prioritization of conflicts is a well-known phenomenon in aviation. A prioritization between aircraft has been made based on various criteria such as current velocity [9], lookahead time [10] and only considering conflicts with aircraft on the right (ROTW) [11]. Nonetheless, these methods have proven to be not so efficient, because they do not prioritize based on the current conflict geometry. It is hard to analytically establish a fixed ruleset which can be applied to a dynamic multi-aircraft conflict with a continuously changing conflict geometry.

Reinforcement learning (RL) could provide a solution by leveraging its ability to approximate the system dynamics and to determine a conflict prioritization strategy based on conflict geometry. The performance of RL models exceeds expert human-level performance on strategic games. For example, a state-of-the-art agent called AlphaGo was able to beat the world champion in Go [12] and more recently, the Alphastar program defeated a top professional player in Starcraft II [13]. This research aims to construct a conflict prioritization policy which will be applied to an existing CR method with RL. The RL framework consists of an actor-critic network which implements Proximal Policy Optimization (PPO) [14]. PPO has shown promising results for multi-agent collision avoidance algorithms in aviation [15], [16] and robotics [17].

The simulations required for the training and evaluation of the deep RL model in this article, were performed in the BlueSky ATC simulator [18]. The SSD method was selected as the CR algorithm to apply conflict prioritization to. SSD is selected, because it implements a global solution which often results in a decrease in safety and efficiency due to a saturated solution space. The improvement upon SSD with conflict prioritization is then directly compared with a pairwise solution method, the Modified Voltage Potential (MVP) method [19].

Although MVP does not implement a global solution, it has proven to be very efficient at high traffic densities.

The detection and prioritization of conflicts in aviation is elaborated upon in section II. Subsequently, the reinforcement learning architecture is discussed in section III. The conflict resolution algorithms that are used in this research are explained in section IV. Thereafter, the experiment design is elaborated upon in section V and the results are presented in section VII. Finally, a discussion and conclusion are provided in section VIII and section IX, respectively.

II. Theoretical Background

A. Conflict Detection

In this research, a state-based conflict detection mechanism is utilized for all experimental conditions. This mechanism assumes a linear propagation of the current state of all aircraft involved. Conflict resolution is applied in a 2-dimensional plane which means that only heading and velocity change can be employed for a conflict resolution manoeuvre.

In manned aviation, the separation distance has to be obeyed with respect to all heading angles, thus forming a circle with the aircraft at its center. This circle is called the protective zone with radius R_{pz}. A loss of separation (LoS) for ownship occurs when an intruder penetrates the protected zone. The terminology ownship and intruder is adopted from Ribeiro et al. [20] and denotes the perspective of the conflict situation. The conflict is described from the perspective of ownship, while the other involved aircraft are defined as intruders.

This section further explains the characteristics of a conflict situation, such as visualized in Figure 1, in more detail. The Closest Point of Approach (CPA) is the point in which the distance between two aircraft in motion will reach its minimum value. The distance to Closest Point of Approach (d_{cpa}) is defined as the distance between two aircraft at CPA. The Time to Closest Point of Approach (t_{cpa}) is the time until the closest point of approach is reached. The t_{cpa} and d_{cpa} can be found with Equation 1 and Equation 2, respectively.

$$t_{cpa} = -\frac{\mathbf{d}_{rel} \cdot \mathbf{V}_{rel}}{|\mathbf{V}_{rel}|^2} \tag{1}$$

$$d_{CPA} = \sqrt{\mathbf{d}_{rel}^2 - t_{CPA}^2 \cdot \mathbf{V}_{rel}^2} \tag{2}$$

The time until a loss of separation occurs t_{in}, is computed with Equation 3. A conflict occurs if the current path of an aircraft will result in a loss of separation within a pre-defined lookahead time window τ. Combined with the stated assumptions on linear state propagation, a conflict occurs when $t_{in} \leq \tau$ and $d_{cpa} < R_{PZ}$ [20]. The term t_{in} can also be referred to as the time to loss of separation t_{LoS}.

$$t_{in} = t_{CPA} - \frac{\sqrt{R_{PZ}^2 - d_{CPA}^2}}{\mathbf{V}_{rel}} \tag{3}$$

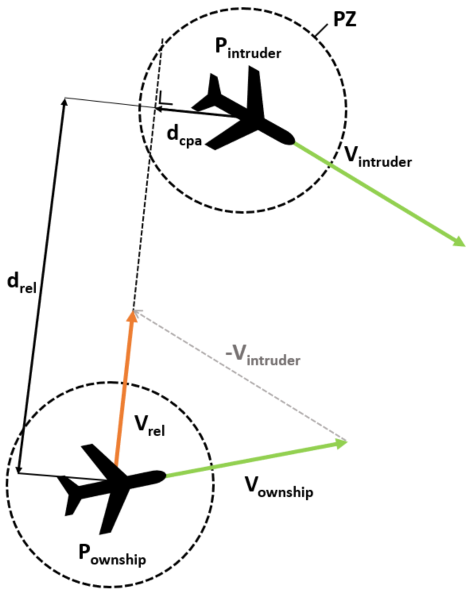


Fig. 1: Description of relation between $d_{cpa}, d_{rel}, v_{rel}$ and positions and velocities of ownship and intruder. Adapted from Ribeiro et al. [20].

B. Prioritization of Intruders

A common approach to apply prioritization in collision avoidance is a centralized approach in which aircraft are assigned a low or high priority based on parameters such as current velocity [9]. Aircraft with low priority should avoid aircraft with a high priority. The right-of-way (ROW) rules as part of the Rules of the Air (RotA) are perhaps the most famous example of prioritization in conflict resolution [11]. The two rules most relevant for this research are for converging and head-on conflicts. For converging conflicts, the aircraft that comes from the right has the Right-of-way and does not need to alter its flight path. The other aircraft has to perform a conflict avoidance manoeuvre. For a head-on conflict, both aircraft have to avoid to the right. It was found that following the RotA did not always reduce the complexity in airspace operations. In a pairwise conflict, the responsibility for maintaining safe separation is appointed to one aircraft while it is often more advantageous if both aircraft perform a conflict resolution manoeuvre to solve the conflict.

Conflict prioritization has also been applied to the SSD algorithm [10]. A conflict prioritization strategy was constructed based on a fixed set of rules. In order of importance, the rules were based on τ, t_{LoS}, d_{LoS} and d_{cpa}. A sequential approach was taken, which means that a conflict rule with a higher priority was selected if the current priority rules still resulted in a completely saturated solution space, and thus no feasible solution. The approach showed improvement over the original SSD method. The major limitations of the research were that only a limited number of rules and a limited number of combinations of those rules were evaluated. Moreover, the rules were not dependent on the conflict geometry.

Conflict prioritization in this research is represented by a binary decision at each simulation timestep. When a conflict is encountered, each aircraft for which state information is available (dependent on ADS-B constraints [21]) is either incorporated in the conflict resolution or completely ignored.

III. Reinforcement Learning Method

A. Markov Decision Process

On an agent level, the mathematical framework used to describe the RL problem of this research is a Markov Decision Process (MDP). An MDP is a mathematical framework in sequential decision making [22]. An MDP can formally be described with the tuple $\langle\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \lambda\rangle$, where $\mathcal{S}$ is the set of all states, $\mathcal{A}$ the set of all actions, $\mathcal{T}$ denotes the set of transition probabilities (the probability that action a_t in state s_t results in state s_{t+1}), $\mathcal{R}$ the set of all rewards and γ represent the *discount factor*.

At time t an agent observes state s_t of the system and takes action $a_t \in \mathcal{A}$. Applying action a_t to the environment transforms the system to state $s_{t+1} \in \mathcal{S}$ and the agent receives reward $r_{t+1} \in \mathcal{R}$. The agent samples its actions from the *policy* $\pi(a|s)$. The policy tries to find the optimal action a based on state s. The goal of the agent is to maximize the discounted reward G_t which is computed according to Equation 4 with $0 \leq \gamma \leq 1$. The discount factor γ reduces the importance of future rewards and enhances the influence of the more current rewards.

$$G_t = \sum_{t=0}^{\infty} \gamma^t r_t, \tag{4}$$

B. RL Method

The Proximal Policy Optimization (PPO) network architecture is adopted for this research [14]. PPO is a policy gradient method where the policy is updated in a controlled way to limit the difference between the old and new policy. Setting a constraint on the size of the policy update, increases the stability of the RL algorithm. PPO has been successfully implemented in a multi-agent collision avoidance setting [15]–[17]. For this research, PPO is selected based on four criteria:

- **Robustness**: The PPO algorithm inherently supports robustness due to the design of the PPO architecture which limits the variance between the old and new policies. The PPO model showed state-of-the art performance with limited hyperparameter tuning in cooperative multi-agent environments [23].
- **Experience Replay**: PPO does not rely on experience replay for convergence which is advantageous, because evolving behaviour of agents makes the environment non-stationary.
- **Hyperparameters**: Tuning of the hyperparameters of a RL algorithm is a time-consuming task and of utmost importance; slight adjustments in the hyperparameter could transform a divergent algorithm into a convergent algorithm. The PPO algorithm is robust for hyperparameter settings [23].
- **Action Space**: In this research, the output of the RL algorithm will be binary. As seen from one agent, the algorithm will decide which intruders it considers (1) and which intruders it ignores (0). The PPO algorithm allows for a discrete action space and performs well on it [24].

The PPO algorithm is implemented with an actor-critic network. In an actor-critic network, the policy-network is the *actor* and determines which action to choose based on the state. The value-network is the *critic*. Since the value function is used when determining the policy gradient and thus used for policy updates, the critic aids in the policy update [22].

Various implementations of PPO exists. In this research, the implementation of PPO with a clipped objective function is implemented. The clipped objective function L^{CLIP} can be found in Equation 5. The policy update r_t is clipped by (1 + ϵ) and (1 - ϵ) for a positive advantage and negative advantage A_t, respectively [14], with ϵ as a hyperparameter. The policy update r_t is defined as the ratio between the old and new policy or $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{Old}}}(a_t|s_t)}$. The actor policy is denoted by π_θ with θ as the model parameters.

$$\mathrm{L}^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min\left(r_t(\theta)\hat{A}_t, \operatorname{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \tag{5}$$

C. Multi-Agent RL

Multi-Agent Reinforcement Learning (MARL) is a branch within RL in which a system of autonomous agents is simultaneously interacting in a common environment [25]. In this research, multiple agents simultaneously apply conflict prioritization, which thus identifies as a multi-agent setting. The multi-agent system in which aircraft apply conflict resolution is cooperative [26]. The agents pursue a common goal in a cooperative environment. They assist each other to maximize a global reward. The system therefore employs a shared reward function.

The domain of multi-agent RL introduces new problems due to the increased complexity of interacting agents while they are evolving. The main challenges are the nonstationarity of the environment [25] and the curse of dimensionality [27]. To tackle these problems, this research adopts the paradigm of "*Centralized Learning with Decentralized Execution*". The agents sample actions based on their own observations (decentralized), but the policy is updated with a centralized value function based on experience of all agents (centralized). Experience consists of states, actions and rewards. For ownship, the input of the centralized value function consists of its own observation vector plus all observations and actions of the observed intruding agents. The environment becomes stationary for ownship by including the observations and actions of the other intruding aircraft. Furthermore, to deal with the scalability, a *shared policy* is implemented. With a *shared policy*, all agents sample from the same policy reducing the number of trainable parameters and thus the training time. The actor-critic network architecture is visualized in Figure 2.

D. State Formulation

Selecting the right variables for the formation of the state vector of the reinforcement learning algorithm is a difficult

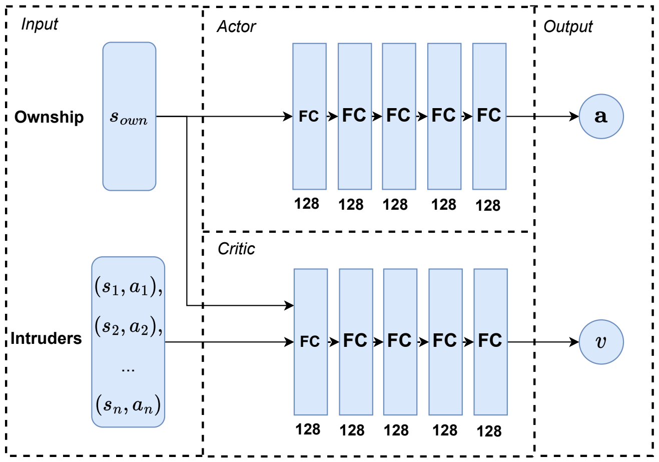


Fig. 2: Schematic overview of actor-critic network architecture with centralized value function for a single time step. The input of the actor model consists of the ownship state s_{own}. The states and actions of intruding aircraft 1 through n are combined with the the state of ownship to form the input of the critic network. Each fully connected (FC) layer in both the actor and critic network consists of 128 hidden nodes. The output of the actor and critic model consists of the action vector **a** and value function estimation *v*, respectively

task. Adding a dimension to the observation space causes the computational complexity to exponentially increase [27]. Thus, it is beneficial to minimize the size of the observation vector.

The t_{cpa} parameter is selected to provide an indicator on how imminent a LoS is with an intruding aircraft. The lat_{rel} and lon_{rel} denote the relative latitude and longitude between ownship and an intruder as seen from the body-fixed coordinate system of ownship. A combination of lat_{rel} and lon_{rel} can provide ownship with both a relative position of the intruder and a distance to the intruder D. The distance D can provide further context to t_{cpa}. The state vector s_t^i of observed aircraft i at time t, is defined according to Equation 6. The full definition of the state vector $\mathbf{s}_t$ can be found in Equation 7 in which n represents the number of observed ac. The size of the observation vector can be computed by multiplying the number of observed aircraft n with the number of features per aircraft. In this research n equals 10 and the number of features 3.

$$s_t^i = \begin{Bmatrix} tcpa_t^i \\ lat_{rel_t}^i \\ lon_{rel_t}^i \end{Bmatrix} \quad (6) \qquad \mathbf{s}_t = \begin{Bmatrix} s_t^1 \\ s_t^2 \\ . \\ . \\ s_t^n \end{Bmatrix} \quad (7)$$

The number of observed aircraft n was set at 10 to stimulate a highly saturated solution space and allow for cooperative behaviour whilst limiting the size of the actor-critic model. The limited model size allows the RL model to train within a practical amount of time. The number of aircraft in the experiments on which the RL model is trained, is limited to 11 such that each aircraft can observe 10 intruders and has information on all aircraft that simultaneously operate in the same airspace.

E. Reward Formulation

The reward function $r(s_t)$ for a single aircraft can be found in Equation 8. A negative reward is given when an intrusion is first encountered.

$$r(s_t) = \begin{cases} -10 & \text{Loss of Separation occurs} \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

F. Action Space

The action vector should be able to individually include (1) or exclude (0) an intruder from the avoidance manoeuvre calculation. The action vector thus consists of binary variables. For each observed aircraft i action $a_i \in \{0, 1\}$. The complete action vector at time t, $\mathbf{a}_t$ is given by Equation 9. The last action in the action vector is referred to as a_{all}. If a_{all} equals 1, all intruders are included in the computation of the of the avoidance manoeuvre. In this manner, the model can opt for a non-prioritized conflict avoidance manoeuvre. In principle, adopting a solution which prevents conflicts with all surrounding intruders will prevent all intrusions. Such a solution should be chosen when it does result in a quick, safe solution.

$$\mathbf{a}_t = \begin{Bmatrix} a_1 \\ a_2 \\ . \\ . \\ a_n \\ a_{all} \end{Bmatrix} \quad (9)$$

IV. Conflict Resolution Algorithms

The proposed RL framework is applied to a decentralized CR method in this research. This section first explains the concept of velocity obstacles, which is the base of the CR algorithm herein used. Thereafter, the Solution Space Diagram (SSD) and Modified Voltage Potential (MVP) algorithms are described.

A. Velocity Obstacles

The idea of implementing velocity obstacles for 2-dimensional motion planning and obstacle avoidance was introduced in 1998 [28]. The concept of velocity obstacles is explained with aid of Figure 3a. The collision cone between ownship and intruder is denoted with CC. Assuming that the current state is linearly propagated, any relative velocity outside of collision cone CC guarantees a collision-free path for ownship and intruder [28]. Subsequently, the collision cone can be transformed from the relative velocity plane to the absolute velocity plane. This simplifies the conflict resolution process since avoidance maneuvers can be planned and executed in the absolute velocity plane. The transformation can be performed according to Equation 10 in which $\oplus$ is the

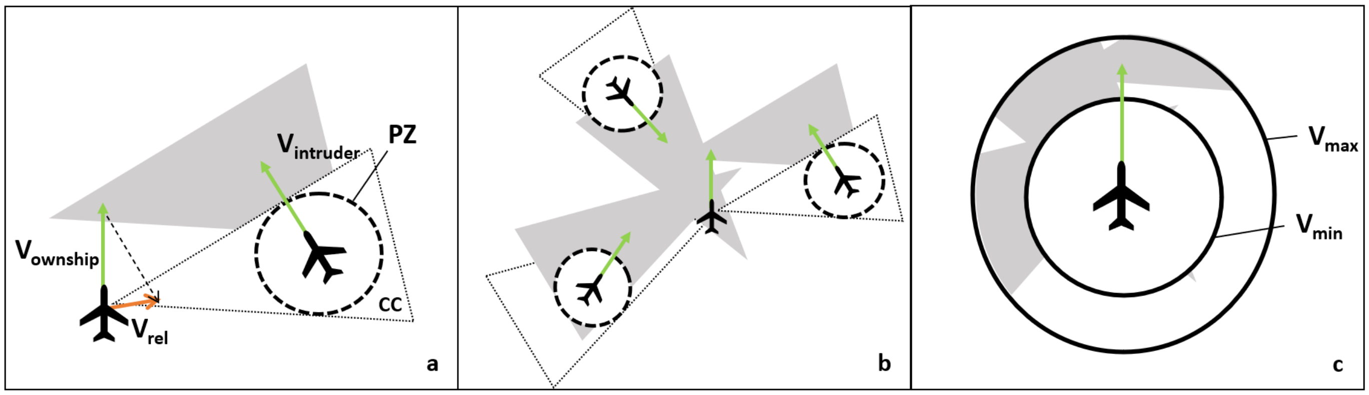


Fig. 3: Construction of velocity obstacle of a single intruding aircraft (a). Visualization of multiple velocity obstacles caused by concurrent intruders (b). The velocity obstacles of the intruding aircraft are combined with the performance limits to form the SSD (c) [7].

Minkowsky vector sum operator. The transformed collision cone is called a velocity obstacle.

$$VO = CC_{A,B} \oplus \mathbf{V_B} \tag{10}$$

B. Solution Space Diagram

The first concept similar to the SSD as implemented by Balasooriyan [7], was introduced by Van Dam et al. as ”the state vector envelope”. The state vector envelope has served several other purposes [29]–[31] before it was implemented as a conflict resolution method [7]. This research implements the SSD in a similar fashion as Balasooriyan [7]. Therefore, the construction process will be similar and is explained with Figure 3. As explained before, Figure 3a displays the construction process of one VO belonging to one intruding aircraft. In Figure 3b, ownship is displayed with three intruding aircraft and their respective collision cones and velocity obstacles. Finally, in Figure 3c, the velocity obstacles are encapsulated by the performance limits of the aircraft to synthesize the SSD in its final form.

If the velocity vector of ownship is located within one of the VOs, the ownship is in a conflict. Therefore, the union of the 3 VOs is called the set of Forbidden Velocities (FV) and can be found with Equation 11. In Figure 3c the set of reachable velocities (RV) is displayed and is based on the performance limits of the aircraft. The set of RV is denoted by the area between two circles. The inner circle has radius V_{min} and the outer circle has radius V_{max}. Assuming that no other regulatory constraints are imposed, an aircraft is able to turn to any desired heading hence the circular shape (for every heading angle, the aircraft at least flies at least at V_{min} and maximally at V_{max}).

$$\mathrm{FV} = \bigcup_{i=1}^{N} \mathrm{VO}_i \tag{11}$$

Division of the RV into the Allowed Reachable Velocities (ARV) and Forbidden Reachable Velocities (FRV) is the final step to synthesize the SSD. The FRV and ARV can be found with Equation 12 and Equation 13, respectively.

$$\mathrm{FRV} = \mathrm{RV} \cap \mathrm{FV} \tag{12}$$

$$\mathrm{ARV} = \mathrm{RV} \cap \mathrm{FV}^C \tag{13}$$

C. SSD in Conflict Resolution

The SSD diagram provides a set of allowable reachable velocities. To utilize the SSD diagram as a conflict resolution method, a methodology has to be established to select the most optimal desired velocity vector from the set of allowable reachable velocities. The shortest path out rule showed the most promising results in terms of safety, stability and efficiency [7] and will thus be used in this research. The shortest path out rule can be seen in Figure 4 which is a zoomed in version of Figure 3c. Conflict prioritization can be relevant for the SSD algorithm, because it implements a joint solution. A joint solution causes quick saturation of the solution space at high traffic densities which in the extreme could result in no feasible solution.

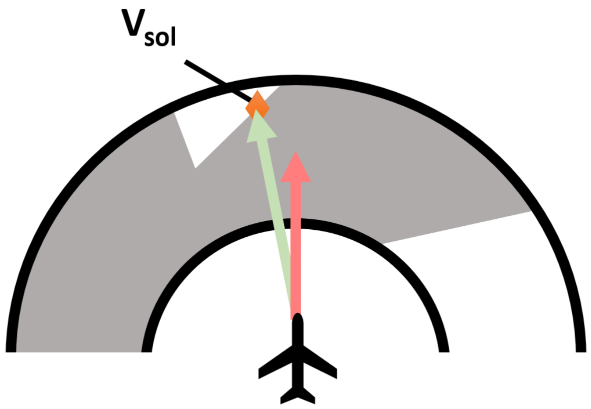


Fig. 4: Current velocity solution vector (red) is in the set of FRV. With conflict resolution, the desired velocity solution vector (green) is computed by applying the shortest path out rule to the SSD

D. SSD in Conflict Prioritization

This research will apply conflict prioritization to the SSD method. For the SSD algorithm, this translates to inclusion/exclusion of VOs, belonging to the intruding aircraft, in the SSD. The influence of conflict prioritization on the SSD is illustrated in Figure 5. The velocity vector in Figure 5a is in the FRV, which means that the current state results in a collision. Conflict resolution is thus required. When all 3 intruders are taken into account, the aircraft will adapt its velocity vector to V_{sol}, which is found with the shortest path out rule. Due to the highly saturated solution space of the SSD, V_{sol} requires the aircraft to make a sharp turn of almost 180 degrees. The right turn results in an intrusion at time t_2 as can be seen in Figure 5b.

In Figure 5c, one intruding aircraft is not taken into account by deactivating its VO. In the new situation, ownship will continue to fly straight, resulting in Figure 5d. No intrusions occur in the new situation. The initial conflict of ownship is partly solved by the conflict avoidance maneuver of the intruding aircraft. Ownship can continue to fly straight and let the intruding aircraft completely resolve the conflict or ownship can reactivate the intruding aircraft and assist in the conflict resolution process.

It can be beneficial to temporarily exclude a VO from the solution space to prevent an aircraft making a sharp turn. A sharp turn would result in a significant deviation from its original flight path, from which the aircraft has to recover. The aircraft cannot instantaneously change its heading. Large heading changes take time, during which it is not guaranteed that ownship will not encounter an intrusion.

When the solution space is highly saturated, intruding aircraft that for example are relatively far away from ownship, seem like a good candidate to deactivate. The influence of ignoring an intruding aircraft can vary. If the velocity solution vector remains similar, exclusion of the VO has no effect. However, when the velocity solution vector does change, ownship essentially resolves towards a velocity solution vector in the FRV. In case the previously disabled intruder does not change its path, it does mean that the conflict with this intruder will not be resolved and the time to loss of separation will decrease. This is not necessarily negative when the new state of the ownship (who has now resolved conflicts with other intruders) may lead to a more efficient resolution manoeuvre for this remaining intruder. Nevertheless, we do expect the intruder to change its state in order to avoid LoS with the ownship. It may even be that the CR manoeuvre performed by the intruder to avoid ownship is sufficient to resolve the conflict, as exemplified in Figure 5.

E. Modified Voltage Potential

The MVP [19] method is part of the group of force field algorithms, in which a parallel is made between aircraft and electrically charged particles. The idea behind a force field algorithm, is that all aircraft are negatively charged which creates a repulsive force between them which should facilitate safe separation. In Figure 6, a conflict between ownship and intruder is visualized. The goal of the MVP method is to "push" the location of the closest point of approach outside of the protected zone in the same direction as the d_{cpa} vector, to resolve the conflict. This can be accomplished by addition of the V_{MVP} solution vector to V_{rel}. The MVP solution vector is always perpendicular to V_{rel}. The V_{opt} vector indicates the shortest path out solution from the collision cone, which is the geometrically optimal way to solve the conflict, resulting in minimal path deviation [8]. In a multi-aircraft conflict, the MVP method computes its velocity solution vector by separately solving each conflict and subsequent addition of all individual velocity solution vectors.

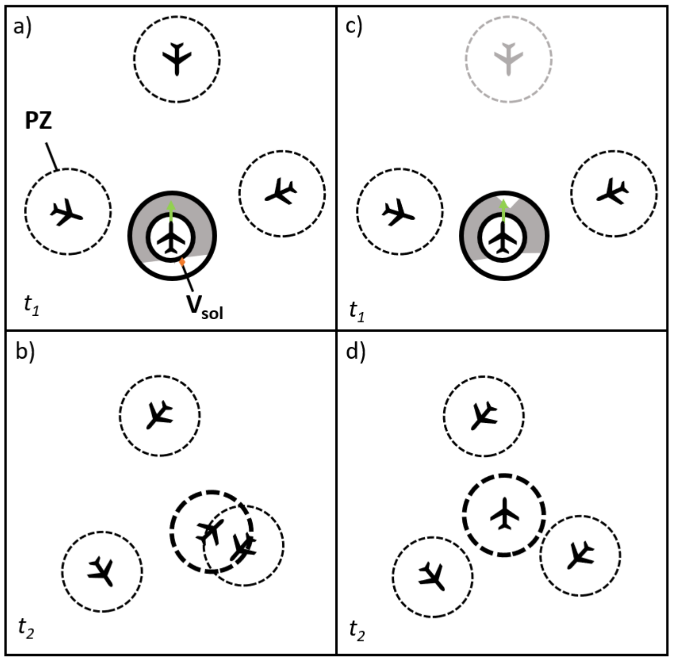


Fig. 5: The left two images visualize the progression of a conflict geometry when ownship does not apply conflict prioritization at time t_1 (a) and t_2 (b). The two images on the right show the same conflict geometry when conflict prioritization is applied to ownship at time t_1 (c) and t_2 (d).

In section VII, MVP and SSD will be directly compared to relate the behaviour of pairwise-summed and joint resolution approaches. In a high traffic density, pairwise-summed methods such as MVP tend to induce more secondary conflicts during conflict resolution. The secondary conflicts can be beneficial, because they cause a redistribution of the traffic. This creates space for new resolution manoeuvres, which were not apparent before. Consequently, its performance tends to be superior in terms of intrusions, compared to a joint resolution approach. With SSD, an increase in traffic density, will likely result in a decrease of the available solution space. SSD does however tends to result in fewer conflicts [20], being better at conflict prevention. Thus, it is interesting to see if conflict prioritization can adapt the characteristics of SSD such that it becomes better at dispersion of traffic in multi-actor conflict situations, while remaining proficient in conflict prevention.

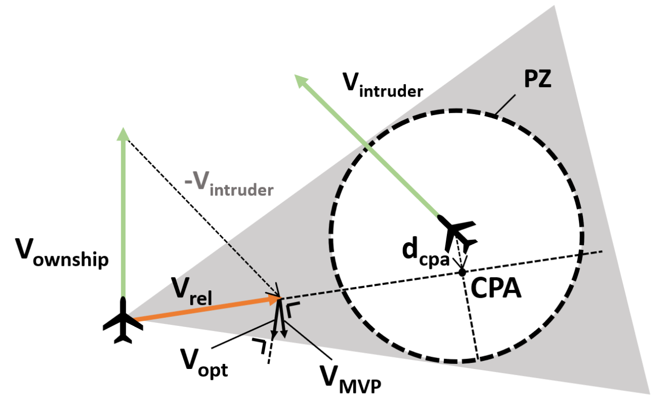


Fig. 6: For the given conflict geometry, the velocity solution vector found by MVP is indicated with V_{MVP} [19]. The geometrically optimal solution V_{opt} [8] is also provided.

V. Experiment Design

A. Simulation Environment

The simulations are performed in the BlueSky environment. BlueSky is a simulator which is established to facilitate a general platform on which ATC research can be performed. This allows for better comparison of ATC research [18]. A great advantage of the BlueSky system is that it implements a server-client architecture, which enables parallel running of experiments on separate CPU nodes, thus several episodes can run simultaneously.

B. Training Architecture

Multi-agent reinforcement learning is a computationally expensive process. Maximizing the computational capacity of the hardware utilized to train the model is beneficial, since it reduces the training time. Ray is a distributed framework which empowers systems to cope with intensive parallel simulations of experiments [32]. RLlib is part of the Ray project and is a RL library which supports high scalability [33]. With RLlib, the BlueSky server-client interface can be leveraged for parallel collecting of experience. For every simulation environment, a client is generated which connects to a simulation node in its own thread by a multi-agent environment. Every client can communicate with the server without interfering with the other clients.

All environments first sample actions from the same actor model. Subsequently, BlueSky is run for one simulation time step dt, with conflict prioritization as determined by the sampled actions. The simulation time step dt equals 2 seconds in this research. The new state of BlueSky is processed in the multi-agent environment interface and transformed into a new set of observations and rewards. The observations, actions and rewards are stored in the experience buffer. Once the experience buffer size is equal to the predefined train batch size, the experiments are resetted and the experience is used to update the actor and critic network. An important step is the postprocessing of the experience. During postprocessing of the experience, the actions and observations for every time step are shared among all agents. This is required to form the input for the centralized value function. The PPO model trains the policy in an on-policy fashion. Therefore, the experience buffer is emptied after the policy update. An overview of the RL framework can be found in Figure 7.

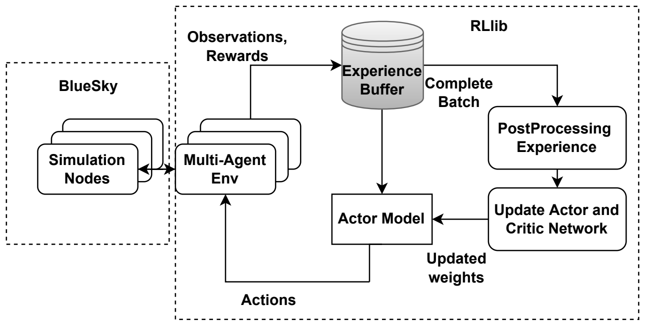


Fig. 7: Schematic overview of multi-agent RL training architecture and visualization of integration RLlib framework with BlueSky.

C. Challenges of Model Training

The first challenge when designing the traffic scenario was the sparsity of the rewards. The reward function (Equation 8) provides sparse rewards, because a reward is only nonzero when a LoS occurs. Achieving a convergent RL model with sparse rewards is an arduous task, because the RL model receives limited feedback from the environment which does not enable the model to train in a practical amount of time. Furthermore, between take-off and landing, an aircraft is not continuously in conflict with one or more intruding aircraft. For a considerable amount of the time, it will be conflict free. If the aircraft is not in conflict, conflict prioritization has no effect, because including/excluding of intruders does not influence the velocity and heading of the aircraft. From that perspective, it would be ideal to have many conflicts and intrusion per unit of simulation time, because that would reduce the sparsity of the rewards and increase the influence of the RL model on the conflict situation. However, if the traffic scenario becomes too complex, it becomes (almost) impossible to solve, which will reduce the stability of the learning process. Thus it was decided, instead, to have the RL model resolve randomized conflict geometries during training. The traffic scenario is designed such that the number and placement of aircraft facilitate multi-aircraft conflict situations in which the original SSD method fails, while limiting the variability, complexity and length of the scenarios to allow for a convergent model.

D. Experiment Specifications

This section will first explain the traffic scenarios used for model training. Thereafter, the more generalized traffic scenario for model evaluation is elaborated upon. A schematic example of a training scenario at t_0 can be found in Figure 8. The experimental area is a square. The total number of aircraft per scenario is limited to 11. This ensures that all intruding aircraft can be incorporated in the state vector of ownship as was explained in subsection III-D. The total experimental area is divided into four smaller squares. To induce conflicts and intrusions and to limit the variability between scenarios, the initial heading and position of the aircraft are not completely randomized. The initial positions of the aircraft are generated such that the number of aircraft in each of the four squares is equal to two or three. Based on the initial position of the aircraft, the target waypoint is selected in the opposing, non-adjacent square as denoted by the blue waypoint marker in Figure 8. At the start of the episode, aircraft are removed such that no aircraft are in LoS. The aircraft are spawned with a true airspeed (TAS) of 458 kts. Every episode has a fixed length of 270 s. The horizontal separation limit is defined by the ICAO [34] and equals 5 nm when radar, ADS-B or MLAT is utilised to determine the aircraft position. The radius of the protected zone R_{pz} is thus set at 5 nm. A lookahead time of 5 minutes is selected for conflict detection in the experiments.

During testing, a more generalized experiment is synthesized which strives to simulate a more operational situation in which both safety and efficiency can be measured. The basic experimental setup is similar to the training scenarios, but there are a few discrepancies:

- The number of aircraft is increased. Within the experimental area, the number of aircraft is kept fixed to ensure a constant traffic density. Experiments are performed with a low, medium and high traffic density in which 30, 50 and 70 aircraft, respectively, are simulated in the experimental area.
- A waypoint is assigned in similar fashion as in the training scenarios. However, a waypoint is now located outside of the experimental area as can be seen by the red waypoint marker in Figure 8. In this manner, aircraft that leave the experimental area can reach their waypoint with relative ease and without encountering many intruders. Aircraft are removed from the simulation once they have reached their waypoint.
- The generalized traffic scenario is focused on the steady-state phase of the experiment rather than the initial, transient phase. The results for the first 15 minutes of an experiment, as the traffic density builds-up to the desired value, are therefore disregarded.
- The number of aircraft is kept fixed in the experimental area, every time an aircraft leaves the area, a new aircraft is spawned. Aircraft are spawned with at least 15 nm to the closest intruding aircraft.

The number of simulated aircraft for all traffic densities is significantly higher than the 11 utilized in the training scenarios. The higher number of aircraft is required to achieve saturation of the solution space in an operational setting and to ensure the aircraft encounter multiple conflict situations before reaching their waypoint. The size of the experimental area is increased to allow for a higher number of simulated aircraft. Once an aircraft leaves the experimental area, the measurements of safety and efficiency continue until the aircraft is removed from the simulation.

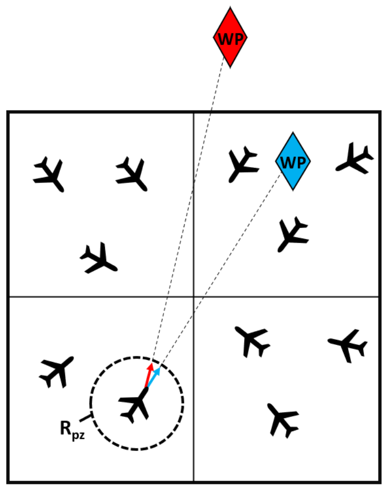


Fig. 8: Schematic representation of the experimental area with 11 aircraft at t_0 for a training scenario. The blue waypoint denotes the waypoint definition in the training scenarios. The red waypoint, which is located outside of the experimental area, describes the waypoint definition for the generalized traffic scenario.

An overview of the traffic scenario parameters for the training and generalized traffic scenario definition can be found in Table I.

TABLE I: Parameters of training and generalized traffic scenario

	Parameter	Value	Unit
Training	lat_{min},lat_{max}	$[-0.35, 0.35]$	°
	lon_{min},lon_{max}	$[-0.35, 0.35]$	°
	Experimental Area	1770	NM^2
	N_{ac}	11	-
	Density	$6.21 \cdot 10^{-3}$	AC/NM^2
General	lat_{min},lat_{max}	$[-1, 1]$	°
	lon_{min},lon_{max}	$[-1, 1]$	°
	Experimental Area	14400	NM^2
	$N_{ac_{low}}$	30	-
	$N_{ac_{medium}}$	50	-
	$N_{ac_{high}}$	70	-
Both	AC type	Boeing 747-700	-
	Altitude	FL350 or 35000	ft
	$\text{TAS}_{initial}$	458	kts
	TAS_{min}, TAS_{max}	[400, 458]	kts
	τ	5	min
	R_{pz}	5	nm

E. Independent Variables

During the experiments, a select number of independent variables is varied to change the experimental conditions. An overview of the independent variables can be found in Table II. Each experiment is performed with either SSD, SSD+PRIO or MVP as conflict resolution algorithm. SSD+PRIO is the method that is evaluated, SSD and MVP are used as baselines for comparison. Moreover, low, medium and high traffic densities are implemented.

TABLE II: Overview of independent variables in the experiments. Each experiment is performed with a specific CR method and density.

Parameter	Value	Unit
CR method	SSD	[-]
CR method	SSD + PRIO	[-]
CR method	MVP	[-]
Density$_{low}$	2.08 $\cdot 10^{-3}$	AC/NM^2
Density$_{medium}$	3.47 $\cdot 10^{-3}$	AC/NM^2
Density$_{high}$	4.86 $\cdot 10^{-3}$	AC/NM^2

F. Dependent Variables

In this section, the definitions are provided for the variables that are measured during the experiments to assess the safety and efficiency of the conflict resolution method. The metrics are based on previous research [7], [35]–[37]. An overview of the dependent variables can be found in Table III.

Safety concerns adequate separation between aircraft. Safe separation can be expressed in terms of N_{LoS} and N_{conf}. The terms LoS and intrusion indicate the same phenomenon and can be interchanged. An intrusion does not directly imply a collision. Differentiation between various intrusions is accomplished by two parameters: the severity of LoS LoS_{sev} and the duration of LoS T_{LoS}. The LoS_{sev} can be computed with Equation 14 in which R represent the radius of the protected zone.

$$LoS_{sev} = \frac{R_{pz} - d_{CPA}}{R} \tag{14}$$

The term T_{LoS} indicates the duration of a single LoS. The total time an aircraft spent in conflict is denotes by T_{inconf}.

Another objective for which conflict resolution methods can be optimized is efficiency. Efficiency is measured with the flight time T and the length of the traversed flight path D.

TABLE III: Overview of dependent variables used in the assessment of SSD algorithm as a CD&R method, adapted from Balasooriyan [7]

Variable	Type	Description
N_{conf}	Safety	Number of conflicts
N_{LoS}	Safety	Number of losses of separations
LoS_{sev}	Safety	Loss of separation severity
T_{LoS}	Safety	Duration of a loss of separation
T_{inconf}	Safety	Total time in conflict per ac
T	Efficiency	Duration of flight
D	Efficiency	Travelled distance

VI. Experiment Hypotheses

It is hypothesized that the RL model is able to learn optimal conflict prioritization decisions that decrease the number of intrusions in all 20 scenarios, but it remains uncertain whether the learned behaviour will generalize well to unknown conflict geometries. The number of trained conflicts situations might be too limited for the model to learn behaviour that can also increase efficiency in unseen conflict geometries. When analyzing the train scenarios to investigate how conflict prioritization improved the safety of a scenario, it is thus hypothesized that some solutions can be implemented in a variety of conflict geometries. Other solutions will only be applicable to specific conflict geometries.

Furthermore, it is hypothesized that the aircraft will show cooperative behaviour due to the shared reward function. It is expected that the RL model can make a prioritization based on t_{cpa}, the relative distance and position of the intruders.

In the experiments with the generalized traffic scenario, it is hypothesized that SSD plus a conflict prioritization strategy will reduce the number of intrusions compared to the regular SSD method. Finally, the conflict prioritization strategy shifts the global solution of SSD, when this is not the most efficient solution, to a more pairwise solution, similar to MVP. The performance of the SSD method with conflict prioritization strategy is thus expected to be similar to the MVP method [19].

VII. Results

The results section consists of two parts. In the first part, the choices that the RL model made to reduce the number of intrusions in the 20 training scenarios, are thorougly analyzed. Based on the results of that analysis, a conflict prioritization strategy is synthesized and applied to the regular SSD algorithm. In the second part, the SSD algorithm with a conflict prioritization strategy is compared to the regular SSD and MVP method [19] when utilized in experiments which resemble a more generalized airspace. For the 20 training scenarios, the discussed dependent variables will be limited to the number of intrusions N_{LoS} and the number of conflicts N_{conf}. An extensive comparison with a greater variety of dependent variables will be provided with the results of the generalized airspace experiments.

A. Training Results

The RL model was trained with 20 randomly generated scenarios. At the start of every episode, one of the 20 scenarios is selected as the starting state of the experiment. The mean reward per episode for an increasing number of training steps can be found in Figure 9. The displayed reward is averaged over the last 100 episodes. Figure 9 displays the progress for 470M training steps.

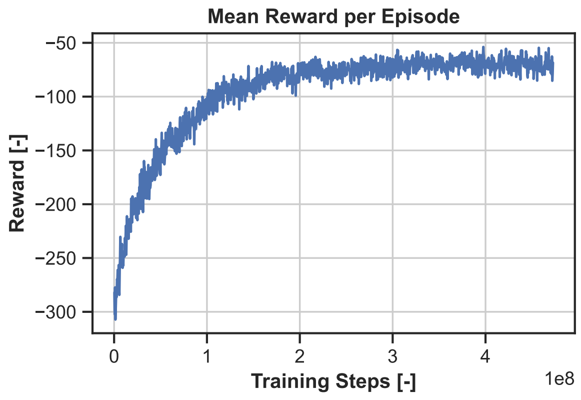


Fig. 9: The evolution of the mean reward per episode during training

An overview of N_{LoS} and N_{conf} per scenario can be found in Figure 10 and Figure 11, respectively. The total

N_{LoS} decreased with 74% from 46 to 12 and the total N_{conf} increased with 6% from 744 to 789.

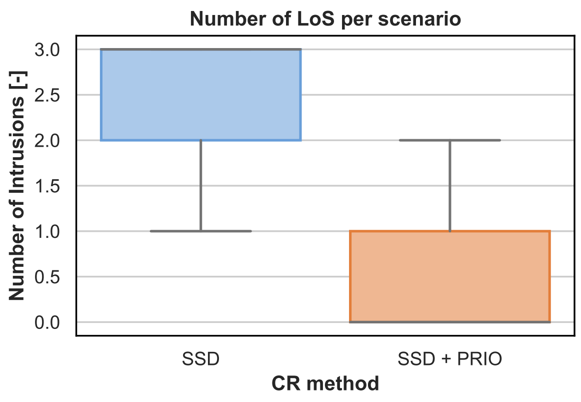


Fig. 10: The number of intrusions per training scenario for the regular SSD method and the SSD method with conflict prioritization (SSD + PRIO)

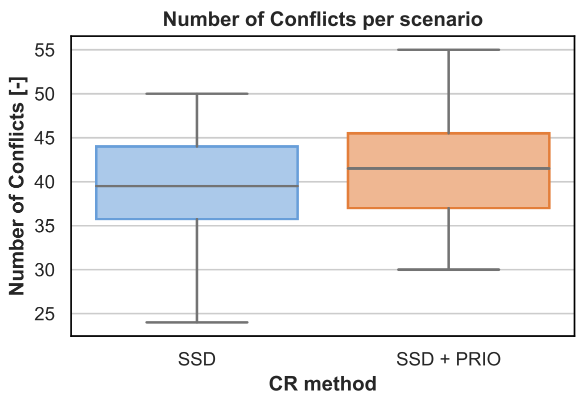


Fig. 11: The number of conflicts per training scenario for the regular SSD method and the SSD method with conflict prioritization (SSD + PRIO)

B. *Improving SSD with Ruleset*

During flight, an aircraft can encounter an infinite number of conflict geometries. Training the RL model on 20 conflict scenarios does not provide sufficient variability to apply the RL model to a generalized traffic scenario. The high number of unknown conflict geometries results in poor model performance. However, from the results in Figure 10 and Figure 11, it can be seen that the model does significantly reduce N_{LoS}. Thus, it is interesting to further investigate per scenario which model choices led to this.

Analyzing the choices of a RL model is not trivial, because the RL model is essentially a black box, it does not provide insight into the reasoning behind the choices that it makes. A methodology to analyze a scenario was constructed and consists of the following steps:

1) **Analysis of Flight Path**: Compare the flight paths of the scenario with and without conflict prioritization. Examine the cases where conflict prioritization resulted in a different flight path.
2) **SSD Comparison**: For the selected aircraft, the SSD is constructed with and without conflict prioritization. The different flight paths can be caused by a different conflict geometry or activation/deactivation of VOs.
3) **Observation Vector Evaluation**: For the aircraft which had a different flight path based on conflict prioritization, the observation vector is analyzed. The goal is to establish a relation between the observation vector variables and the activation/deactivation of intruding aircraft.
4) **Conclusion**: Summarize the found relations per scenario.

The conclusions that resulted from this approach were further investigated. Only a select number of similar conclusions were found in different scenarios and had the potential to generalize well beyond those scenarios. A conflict prioritization strategy was synthesized based on those conclusions.

Intruders need to be included to allow for conflict resolution. Therefore, the requirements for deactivating intruders are more extensive than for activating intruders. The goal of conflict prioritization is to increase the ARV. This is accomplished by only selecting a limited number of aircraft within ADS-B distance. The aircraft should be selected such that enough information on conflict geometry remains present to perform conflict resolution. In this research, a maximum of 10 intruding aircraft can be selected by the CR algorithm for conflict resolution. The found prioritization rules were based on a RL model that could only select up to 10 intruders. With a different number, it is likely that less or more severe prioritization rules would have been found. Selecting a maximum of 10 aircraft also allows for a degree of comparison between the testing and the training results. However, when the experimental setup changes, it is possible that the maximum number of selected aircraft needs to be recalibrated. The aircraft are selected based on the following prioritization strategy:

1) Conflicting Aircraft: The first rule is the prioritization of conflicting aircraft. An example from one of the scenarios from which rule 1 originates, can be found in Figure 12. The position of ownship is set at (0°,0°) as a body-fixed reference frame is implemented in which lon_{rel} and lat_{rel} are measured with respect to ownship. The trajectories of the intruding aircraft are clearly visible. The blue circles are increased in size to denote the change in position of the intruding aircraft as seen by ownship over time. Since it is a body-fixed reference frame, the position of ownship is not altered. The VOs of intruding aircraft AC2 are activated for all but one time step. From Figure 12 and Figure 13, based on t_{cpa} and relative distance, it does not become clear why AC2 is selected. An explanation is provided by Figure 14 which shows the d_{cpa} values of the intruding aircraft. With $d_{cpa} < R_{pz}$ and $t_{cpa} < \tau$, AC2 is found to be a conflicting intruder. The phenomenon of selecting conflicting intruders over non-conflicting intruders occurred in multiple scenarios. Hence, it is converted to a rule. Conflicting aircraft are always selected if they are within ADS-B range.

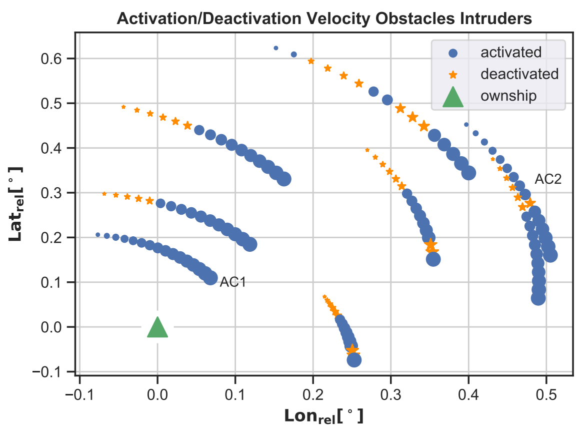


Fig. 12: Conflict scenario in which the activation and deactivation of the velocity obstacles of eight intruding aircraft for ownship are visualized. The increase in size indicates the direction in which the intruders travelled.

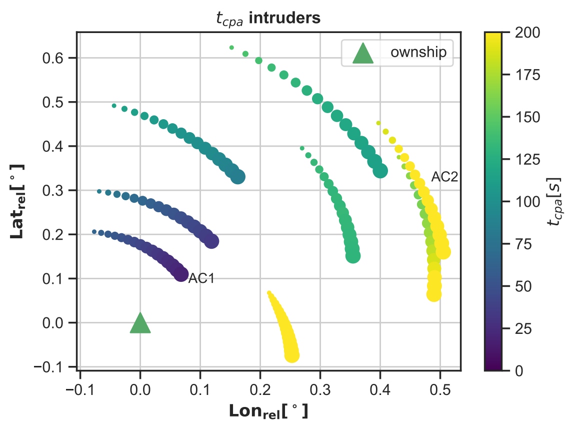


Fig. 13: The trajectories of the eight intruding aircraft are visualized. The color of the data points indicate the time to closest point of approach for every intruder with ownship.

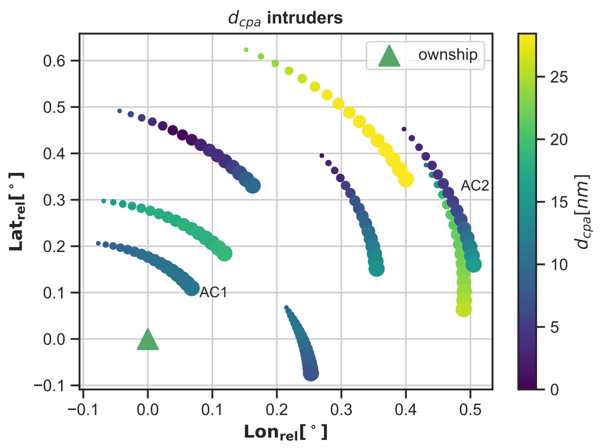


Fig. 14: The relative trajectories of the intruding aircraft are visualized in a body-fixed reference frame with ownship at the origin. The color of the data points indicate the distance to closest point of approach for every intruder with ownship.

2) Closest Aircraft: Moreover, it was observed that intruders with the smallest t_{cpa}, lat_{rel} and lon_{rel} were prioritized. In Figure 12, intruding aircraft AC1 is closest to ownship in terms of relative position and has the lowest t_{cpa} (Figure 13). Intruding aircraft AC1 is the only intruder which is activated for all time steps in which ownship was in conflict. This was expected, because it makes sense that the closest aircraft for which an intrusion is imminent are prioritized over aircraft whose conflicts can be solved at a later point in time. Therefore, after the selection of conflicting aircraft, the aircraft that are closest in terms of relative distance are prioritized. The t_{cpa} parameter is deliberately not included. If an intruder is very close to ownship, but on a parallel trajectory, t_{cpa} is infinite. However, a sudden change in direction of ownship or the intruding aircraft could result in a very close range conflict which justifies the exclusion of t_{cpa}. Aircraft are selected such that combined with the first step, a total up to 10 aircraft are selected.

3) Removal of Aircraft: The final step concerns the deactivation of intruders which are determined to be irrelevant by the RL model. An example is provided with aid of Figure 15, Figure 16 and Figure 17 which showcase the first 18 time steps of a conflict scenario. After 18 time steps, all intruders are activated for the remainder of the conflict scenario. For the first 18 time steps, AC1 is almost fully activated and AC2 and AC3 are mostly deactivated. Again, AC1 is selected as it is the closest aircraft with the highest t_{cpa}. From Figure 17, it becomes clear that AC2 is non-conflicting and AC3 is conflicting. For this specific scenario, activation/deactivation of AC3 has a very minor effect on the ARV. Therefore, the focus of this section is on AC2. Deactivation of a non-conflicting intruding aircraft ($d_{cpa} > R_{pz}$) that has a high t_{cpa} and large relative distance, occurred in multiple scenarios. Therefore, with the incorporation of a safety factor, the final procedure is to deactivate selected intruders that have a $d_{cpa} >$ 15 nm, $D >$ 30 nm and $t_{cpa} >$ 125 s.

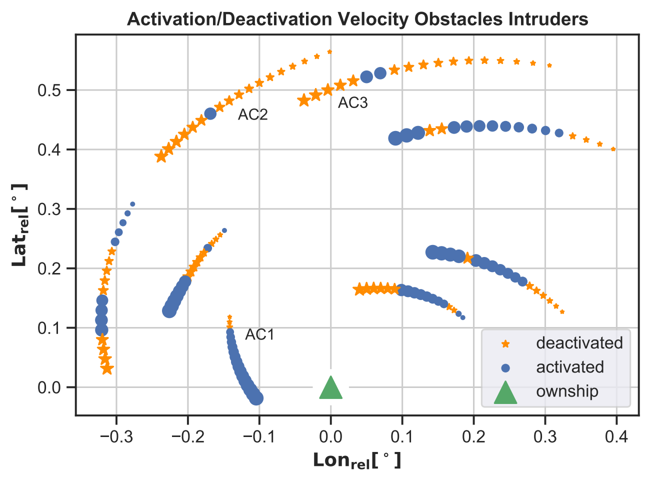


Fig. 15: Conflict scenario in which the activation and deactivation of the velocity obstacles of eight intruding aircraft for ownship are visualized. The increase in size indicates the direction in which the intruders travelled.

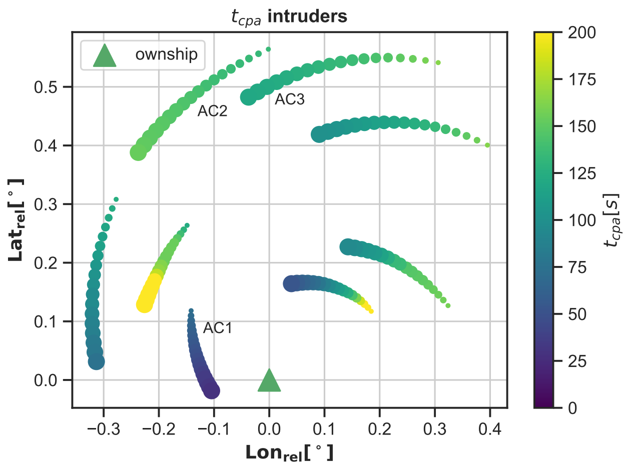


Fig. 16: The relative trajectories of the intruding aircraft are visualized in a body-fixed reference frame with ownship at the origin. The color of the data points indicate the time to closest point of approach for every intruder with ownship.

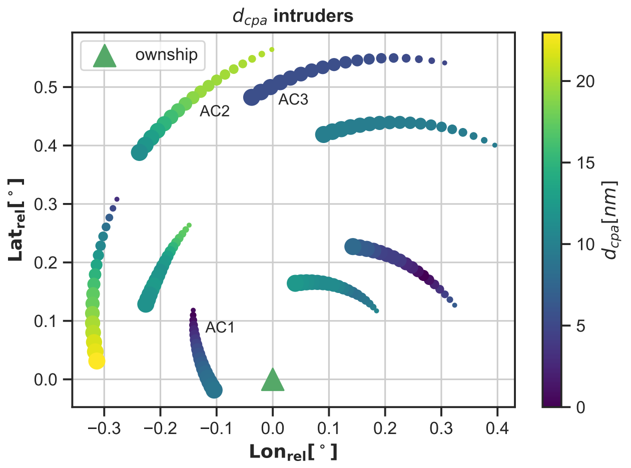


Fig. 17: The relative trajectories of the intruding aircraft are visualized in a body-fixed reference frame with ownship at the origin. The color of the data points indicate the distance to closest point of approach for every intruder with ownship.

To summarize, conflicting aircraft are first selected. The intruders which are closest to ownship are subsequently selected until the total number of intruders equals 10. Finally, intruders can be removed from those 10 selected intruders based on the requirements stated in the previous paragraph.

C. Generalized Traffic Scenario

The experimental results for the generalized traffic scenario are presented in this section. The SSD+PRIO abbreviation utilized in the charts in this section refers to the original SSD method combined with the prioritization strategy as described in subsection VII-B. For each experimental setting, 6 scenarios were simulated for 6 hours. Per experimental setting a specific CR model and traffic density were evaluated. The outliers are not shown in the graphs.

1) Safety: The results for the number of intrusions N_{LoS} can be found in Figure 18. At the low traffic density, the performance of SSD+PRIO results in slightly fewer intrusions than SSD+PRIO, but the results are (almost) comparable. However, at a medium and high traffic density, when PRIO becomes more important, SSD+PRIO significantly outperforms the original SSD method in terms of intrusions. MVP outperforms SSD and SSD+PRIO in all traffic densities.

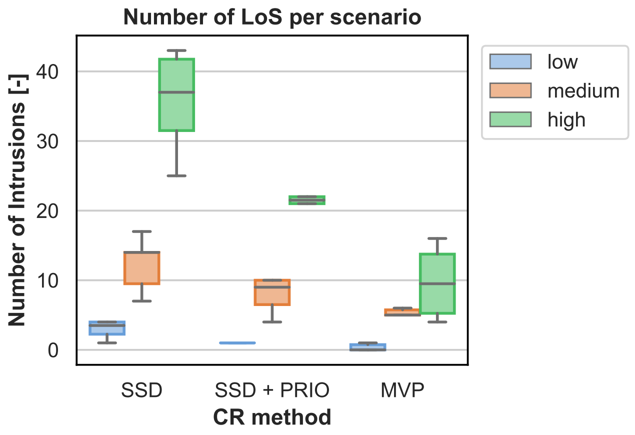


Fig. 18: Number of LoS for low, medium and high traffic densities. The N_{LoS} are summed per experiment, which equals 6 hours of simulation time.

To provide more context to the N_{LoS} numbers, Figure 19 displays the results for LoS_{sev}. It can be seen that the SSD+PRIO method also reduces the LoS_{sev} compared to SSD in all traffic densities. The change in traffic density does not have a great effect on LoS_{sev}. The intrusions that MVP had, are significantly less severe compared to both SSD and SSD+PRIO in a low, medium and high traffic density.

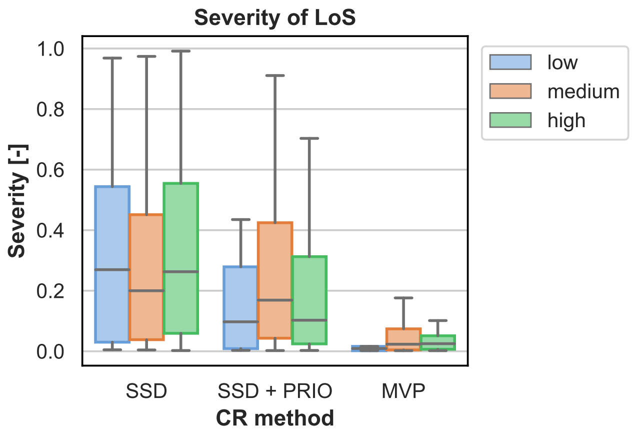


Fig. 19: Severity of losses of separation for low medium and high traffic densities.

The final descriptive parameter for the intrusions is T_{LoS}. The T_{LoS} values for SSD and SSD+PRIO are quite similar. SSD shows a slight increase in T_{LoS} when switching from a low to a high traffic density. The T_{LoS} for MVP is smaller than SSD and SSD+PRIO for all traffic densities.

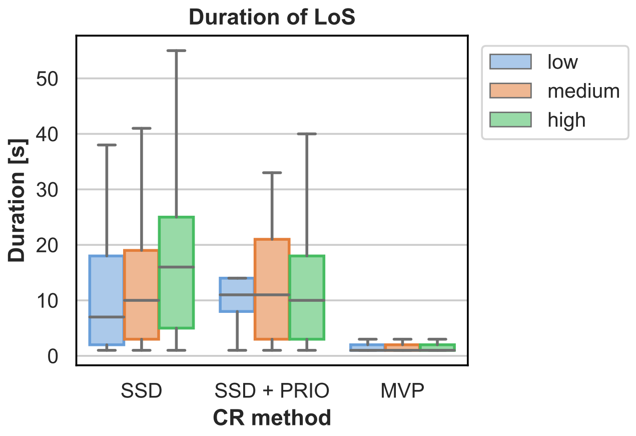


Fig. 20: Duration of LoS for low, medium and high traffic densities.

The dependent variables N_{conf}, T_{conf} and T_{inconf} are based on the conflicts which occurred during the experiments. The results for the number of conflicts N_{conf} is visualized in Figure 21. With all three CR methods, an exponential increase in N_{conf} occurs when the traffic density increases. It should be noted that the number of conflicts for SSD+PRIO is considerably larger compared to SSD. For any traffic density, MVP has the highest number of conflicts.

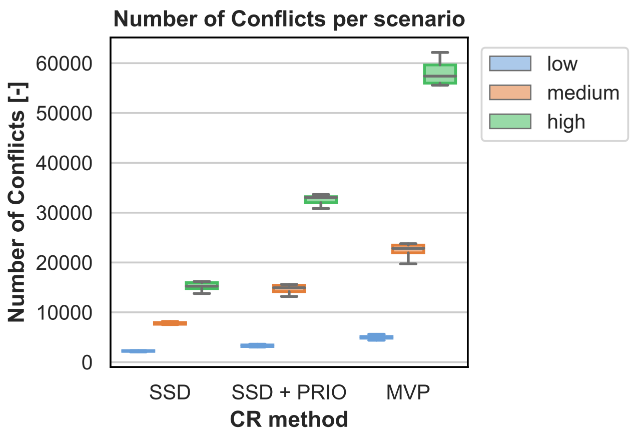


Fig. 21: The number of conflicts for low, medium and high traffic density.

At last, Figure 22 displays the time an aircraft spent in conflict T_{inconf}. Naturally, T_{inconf} shows an increment due to an increase in traffic density for all three CR methods. Interestingly, when using SSD+PRIO, aircraft spent the most time in conflict for all three traffic densities.

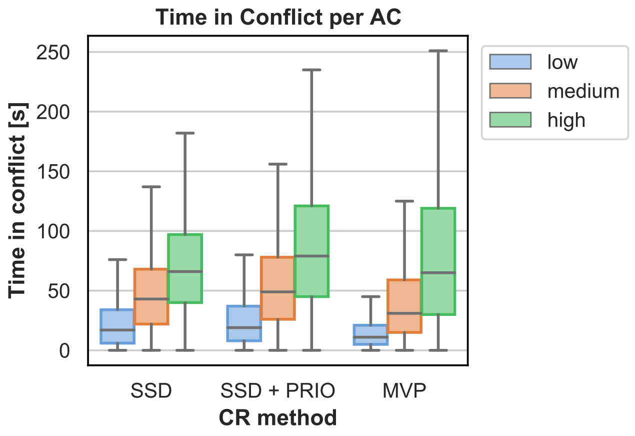


Fig. 22: The time an aircraft spent in conflict for low, medium and high traffic densities.

2) *Efficiency:* The efficiency of a CR method was measured with flight time T and travelled distance D for which the results can be found in Figure 23 and Figure 24, respectively. In a low traffic density, the results for T and D are similar for SSD, SSD+PRIO and MVP. Performance of SSD+PRIO exceeds performance of SSD on both metrics in the medium and high traffic densities. Furthermore, an increment in traffic density had a negative effect on T and D for all 3 CR methods. The increased traffic density evokes more conflict avoidance manoeuvres which cause larger deviations of the aircraft from the nominal route. The MVP method performs best with regard to efficiency. It is interesting that the MVP method excels in efficiency, despite the high number of conflicts. The deviation from nominal flight path is small for every conflict avoidance manoeuvre in MVP. The small conflict avoidance manoeuvres in MVP create a wave-like effect which also helps to reduce the number of intrusions.

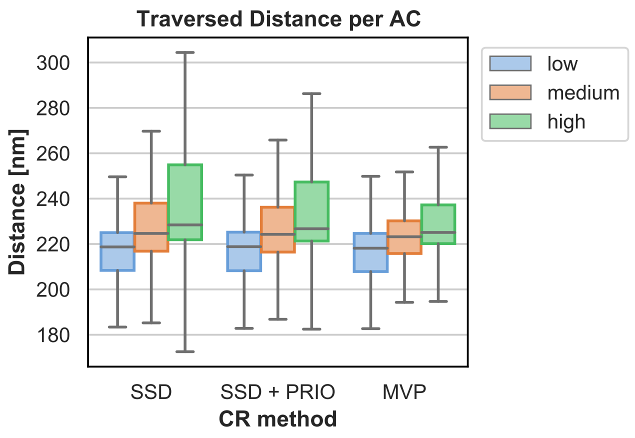


Fig. 23: The travelled distance D per flight for low, medium and high traffic densities.

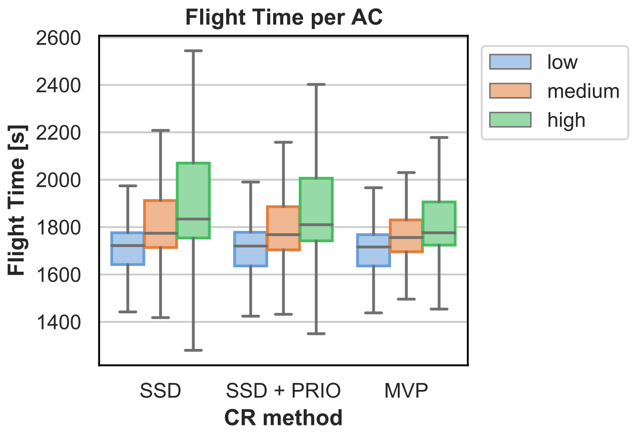


Fig. 24: The flight time T for low, medium and high traffic densities.

VIII. Discussion

The experiments in this researched aimed to improve the safety and efficiency of a decentralized CR method at a high traffic density by applying conflict prioritization with deep RL. Aircraft encounter a multitude of conflict geometries in experiments with a generalized traffic scenario. Establishing a model which can resolve all conflict geometries is impossible. Besides, training on such experiments is very time-consuming and difficult, because of the sparsity of rewards. Consequently, it was decided to train the RL model on a limited number of 20 conflict geometries. The model could have been trained on more scenarios, but that also became unpractical time-wise and still does not provide assurance that the model will perform well on any unseen conflict geometry. As hypothesized, the model was able to successfully reduce the number of intrusions on 20 conflict scenarios.

It was decided that the best approach to leverage the potential of the RL model, was to develop a PRIO strategy based on the intruder selection of the RL model in the training scenarios. The establishment of a PRIO strategy ensures that, during real world operations, conflict prioritization decisions will always be controllable and explainable. The PRIO strategy essentially consists of 3 steps. First, all conflicting aircraft are selected. Subsequently, a number of the closest intruders are also included. Finally, from the selected intruders, aircraft can be removed if they have a high relative distance, t_{cpa} and d_{cpa}. The thresholds for the aforementioned parameters are dependent on the performance limits of the aircraft. Removal of such aircraft further increases the set of ARV. These general rules led to a higher prevention rate of intrusions.

The discussion on the RL facet of this research will be provided in subsection VIII-A. Thereafter, the results of applying the conflict prioritization strategy specifically to SSD, together with a comparison with MVP, are discussed in subsection VIII-B. Recommendations for future work are given in subsection VIII-C.

A. RL Model Settings

The relation between the state vector of the RL model and the selected intruders was thoroughly analyzed. The RL model was able to prioritize intruders with a small t_{cpa}, lat_{rel} and lon_{rel} over intruders with a large t_{cpa}, lat_{rel} and lon_{rel}. The lat_{rel} and lon_{rel} parameters were included to provide both a current distance and relative position of the intruder to ownship. There were however no signs that the relative position was used as a basis for a prioritization strategy. The limited number of conflict scenarios could be a reason for the absence of PRIO rules based on relative position. Training of the RL model in a more generalized traffic scenario with lat_{rel} and lon_{rel}, in which a large number of conflict geometries is encountered, could potentially result in PRIO based on relative position.

Furthermore, lat_{rel} and lon_{rel} did play a role in the identification of nearby intruders which allowed the model to solve conflicts. However, some cases were found during training that did not generalize well to all conflict geometries. For example, an intruding aircraft was prioritized while it had the highest relative distance and t_{cpa}. Sometimes, the RL model simply learned that the selection of an intruder based on a particular state of the environment, resulted in the highest reward. If the model applies similar logic in new conflict situations, it is unlikely that it will result in a reduction of N_{LoS}. The explanation for this is overfitting of the RL model on the training scenarios. To prevent overfitting, the model should thus be trained on a larger amount of conflict geometries. Besides, it may also be that the RL model requires more information from the environment to improve its decisions.

B. SSD with Conflict Prioritization

The results of this research show that the addition of conflict prioritization to the SSD method reduced both the number of intrusions, and the effect of conflict resolution on flight path and time. The difference in N_{LoS} between the SSD and SSD+PRIO method, increases with the traffic density, which was expected. As the traffic density increases, the saturation of the solution space becomes more problematic and the influence of PRIO increases. The SSD+PRIO method has less severe intrusions than the SSD method. The most severe intrusions are often caused by uncoordinated behaviour (aircraft turn into each other in an attempt to avoid each other) [7]. The conflict prioritization strategy does reduce the frequency at which uncoordinated behaviour occurs as can be seen in the reduction in the number of intrusions. However, the current conflict prioritization strategy does not completely eliminate uncoordinated behaviour which explains the LoS_{sev} values for SSD+PRIO. The number, duration and severity of the intrusions were all significantly lower for the MVP method.

Additionally, introduction of the PRIO strategy to the SSD method caused an increment in the number of conflicts. Conflict prioritization disables intruding aircraft to free up solution space in the SSD. If a different velocity solution vector is found in the newly available solution space, this solution is no longer a global solution, because the solution ignores the disabled aircraft. If the aircraft resolves towards a new solution, it is expected that the solution results in new conflicts which subsequently have to be solved.

With PRIO, conflicts are solved with smaller conflict resolution manoeuvres which is facilitated by the increase in ARV. Due to the shorter conflict manoeuvres, ownship is able to more quickly manoeuvre to the desired velocity solution vector. This weakens the unpredictable effect of the dynamic behaviour of the SSD [7]. The effect of the small conflict manoeuvres is visible in the efficiency. The large conflict resolution manoeuvres of the SSD method, cause a great deviation from the nominal flight path. However, SSD+PRIO is still not as efficient as MVP. Thus, albeit manoeuvres are performed based on a smaller number of intruders than with SSD, they still lead to larger deviations than the conflict pairwise solution of MVP.

C. Recommendations

It is proposed to replace lat_{rel} and lon_{rel} with relative distance D and heading with respect to the intruders. Furthermore, it is advised to include information in the observation vector that indicates whether an intruder is a conflicting aircraft or not. This can be done with the d_{cpa} parameter or with a simple binary variable which equals 1 if intruder is conflicting and 0 otherwise. Selecting d_{cpa} or the binary value is a trade-off between complexity and potential effectiveness. The d_{cpa} parameter is more complex to understand for the model, but could potentially provide more information than a simple binary variable.

Finally, with the proposed modifications in the formulation of the RL problem, the PRIO strategy can be further refined by training the RL model on a larger number of more complex conflict geometries.

IX. Conclusion

This work investigated whether conflict prioritization can improve the safety and efficiency of a decentralized CR method in a multi-agent setting with deep RL. The Proximal Policy Optimization (PPO) model was implemented with an actor-critic network and was successfully trained on 20 randomly generated multi-agent conflict scenarios, significantly reducing the number of intrusions.

A conflict prioritization strategy was established based on the optimal actions that the RL model selected. The strategy consists of three steps. The first step is to select all conflicting aircraft. The aircraft that are close by are subsequently incorporated by the CR method. Finally, intruders should be removed from the selection when they are past a safety distance. Naturally, this safety distance is dependent on the performance limits of the operating aircraft. This conflict prioritization strategy significantly decreased intrusions in a generalized airspace while reducing the impact of tactical conflict resolution on the flight path and time.

Future work should extend the training of the RL model to a larger number of different conflict geometries while varying the setup of the RL model. This will likely lead to formulation of additional conflict prioritization rules. Additionally, more elaborate testing of the conflict prioritization strategy of this research is required to verify that it is also beneficial in airspaces with different specifications in which aircraft with different performance limits operate. Moreover, a similar research approach can be applied to different CR methods.

References

[1] EUROCONTROL, *Performance review report an assessment of air traffic management in europe during the calendar year 2018*, 2018.

[2] M. Doole, J. Ellerbroek, and J. Hoekstra, "Estimation of traffic density from drone-based delivery in very low level urban airspace," *Journal of Air Transport Management*, vol. 88, no. June, p. 101 862, 2020, ISSN: 09696997. DOI: 10.1016/j.jairtraman.2020.101862.

[3] SESAR joint Undertaking, *European drones outlook study unlocking the value for europe*, https://www.sesarju.eu/sites/default/files/documents/reports/European_Drones_Outlook_Study_2016.pdf, 2016.

[4] B. Hilburn, "Cognitive complexity in air traffic control: A literature review," *EEC note*, vol. 4, no. 04, pp. 1–80, 2004.

[5] EUROCONTROL, *Model for Task and Job Descriptions of Air Traffic Controllers*. European Air Traffic Control Harmonisation and Integration Programme, 1996.

[6] S. HAO, S. CHENG, and Y. ZHANG, "A multi-aircraft conflict detection and resolution method for 4-dimensional trajectory-based operation," *Chinese Journal of Aeronautics*, vol. 31, no. 7, pp. 1579–1593, Jul. 2018. DOI: 10.1016/j.cja.2018.04.017.

[7] S. Balasooriyan, "Multi-aircraft conflict resolution using velocity obstacles," Master's Thesis, Delft University of Technology, 2017.

[8] J. Ellerbroek, "Airborne conflict resolution in three dimensions"," Ph.D. dissertation, Delft University of Technology, 2013.

[9] H. Emami, F. Derakhshan, and S. Pashazadeh, "A new prioritization method for conflict detection and resolution in air traffic management," *Journal of Emerging Trends in Computing and Information Sciences*, vol. 3, no. 7, pp. 1042–1049, 2012.

[10] L. P. I. da Piedade, "Aircraft conflict prioritization and resolution using the solution space diagram," 2018.

[11] "Right-of-Way Rules: Except Water Operations", *Air Traffic and General Operating Rules*. Federal Aviation Administration Regulation, Title 14, Chap. 1.F, Pt. 91.B, Sec. 91.113, July 2004.

[12] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[13] O. Vinyals, T. Ewalds, S. Bartunov, *et al.*, *Starcraft II: A new challenge for reinforcement learning*, 2017. arXiv: 1708.04782.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal policy optimization algorithms*, 2017. arXiv: 1707.06347.

[15] D. Wang, T. Fan, T. Han, and J. Pan, "A Two-Stage Reinforcement Learning Approach for Multi-UAV Collision Avoidance under Imperfect Sensing," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3098–3105, 2020, ISSN: 23773766. DOI: 10.1109/LRA.2020.2974648.

[16] M. Brittain, X. Yang, and P. Wei, *A deep multi-agent reinforcement learning approach to autonomous separation assurance*, 2020. arXiv: 2003.08353.

[17] T. Fan, P. Long, W. Liu, and J. Pan, *Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios*, 2018. arXiv: 1808.03841.

[18] J. M. Hoekstra and J. Ellerbroek, "Bluesky atc simulator project: An open data and open source approach," in *Proceedings of the 7th international conference on research in air transportation*, FAA/Eurocontrol USA/Europe, vol. 131, 2016, p. 132.

[19] J. M. Hoekstra, R. N. Van Gent, and R. C. Ruigrok, "Designing for safety: The 'free flight' air traffic management concept," *Reliability Engineering and System Safety*, vol. 75, no. 2, pp. 215–232, 2002, ISSN: 09518320. DOI: 10.1016/S0951-8320(01)00096-5.

[20] M. Ribeiro, J. Ellerbroek, and J. Hoekstra, "Review of conflict resolution methods for manned and unmanned aviation," *Aerospace*, vol. 7, no. 6, 2020, ISSN: 22264310. DOI: 10.3390/AEROSPACE7060079.

[21] J. Sun, "Open aircraft performance modeling: Based on an analysis of aircraft surveillance data," Ph.D. dissertation, 2019. DOI: 10 . 4233 / UUID : AF94D535 - 1853 - 4A6C - 8B3F - 77C98A52346A.

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[23] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, *Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks*, 2020. arXiv: 2006.07869.

[24] C. C.-Y. Hsu, C. Mendler-Dünner, and M. Hardt, *Revisiting design choices in proximal policy optimization*, 2020. arXiv: 2009.10897.

[25] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, *Multi-agent actor-critic for mixed cooperative-competitive environments*, 2017. DOI: 10.48550/ARXIV.1706.02275.

[26] P. J. 't Hoen, K. Tuyls, L. Panait, S. Luke, and J. A. La Poutré, "An Overview of Cooperative and Competitive Multiagent Learning BT - Learning and Adaption in Multi-Agent Systems," pp. 1–46, 2006.

[27] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," *Innovations in multi-agent systems and applications-1*, pp. 183–221, 2010.

[28] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998. DOI: 10.36288/roscon2012-900669.

[29] P. Hermes, M. Mulder, M. M. Van Paassen, J. H. Boering, and H. Huisman, "Solution-space-based analysis of the difficulty of aircraft merging tasks," *Journal of Aircraft*, vol. 46, no. 6, pp. 1995–2015, 2009, ISSN: 15333868. DOI: 10.2514/1.42886.

[30] G. A. Mercado Velasco, M. Mulder, and M. M. Van Paassen, "Analysis of air traffic controller workload reduction based on the solution space for the merging task," *AIAA Guidance, Navigation, and Control Conference*, no. August, 2010. DOI: 10.2514/6.2010-7541.

[31] C. Borst, C. Westin, and B. Hilburn, "An investigation into the use of novel conflict detection and resolution automation in air traffic management," *SIDs 2012 - Proceedings of the SESAR Innovation Days*, no. November, 2012.

[32] P. Moritz, R. Nishihara, S. Wang, *et al.*, *Ray: A distributed framework for emerging ai applications*, 2017. arXiv: 1712.05889.

[33] E. Liang, R. Liaw, R. Nishihara, *et al.*, "Ray rllib: A composable and scalable reinforcement learning library," [Online]. Available: https://github.com/ray-project/ray.

[34] ICAO, *Doc 4444 - PANS-ATM, Procedures for Navigation Services – Air Traffic Management*, 16th ed. International Civil Aviation Organisation, 2016.

[35] E. Sunil, J. Ellerbroek, J. Hoekstra, *et al.*, "Analysis of airspace structure and capacity for decentralized separation using fast-time simulations," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 1, pp. 38–51, 2017.

[36] T. Langejan, E. Sunil, J. Ellerbroek, and J. Hoekstra, "Effect of ads-b characteristics on airborne conflict detection and resolution," in *Sixth SESAR Innovation Days, 8th – 10th November 2018*, 2018.

[37] M. Tra, E. Sunil, J. Ellerbroek, and J. Hoekstra, "Modeling the intrinsic safety of unstructured and layered airspace designs," in *Twelfth USA/Europe Air Traffic Management Research and Development Seminar*, 2017.

II

Preliminary Report [already graded]

1 Introduction

In 2019, a staggering amount of 566,031 commercial flights were recorded in the Netherlands alone [11]. Smooth operation of all these flights is enabled by Air Traffic Management (ATM). ATM is a term that embodies various services that are responsible for managing the airspace and air traffic in a safe, economically optimal and efficient manner [39]. ATM has three main components: Air Space Management (ASM), Air Traffic Services (ATS) and Air Traffic Flow Management (ATFM). Part of ATS is Air Traffic Control (ATC) and the main purpose of an Air Traffic Controller (ATCo) is to ensure a safe, efficient and orderly flow of traffic [70]. One of their core tasks which contributes to this process is Conflict Detection and Resolution (CD&R) [18].

In 2018, Europe had 11 million flights and an average annual growth rate of 2.0% was predicted [19]. This increase in air traffic was expected to cause an increment in the workload of the ATCo. The workload of the ATCo is often seen as the bottleneck that would inhibit a further growth in air traffic volume [27]. Various initiatives to evolve from the current ATM system to a new system which can cope with such problems have been established namely SESAR [79] and NextGen [42] for Europe and the United States respectively.

An interesting development parallel to the events that occurred in the manned aviation sector, is the growth and expected growth in number of unmanned aerial vehicles (UAV) employed in a commercial manner. In a case study for the Paris Metropolitan area (12,012 km^2), a conservative estimate was made that the airspace would need to cope with 63,596 delivery drones in 2035 which operate at 0-500 ft above ground level, assuming no regulatory constraints will be imposed. These drones would be employed for small express packages and fast-food meals delivery [13]. A study by SESAR shares this vision and expects around 7 million drones for recreational purposes and around 400,000 for commercial and governmental assignments in 2050. Also, the 2015 EU Aviation Strategy concludes that an increase in the utilization of drone services will take place which could induce major economic growth and produce social benefits [82]. The urge for Unmanned Aircraft System Traffic Management (UTM) becomes evident from such a high expected traffic density and potential commercial and social gains. In Europe, the U-space framework is being developed, which strives to facilitate safe, efficient and secure operations for an extensive number of UAVs in the airspace [80].

A concept which could alleviate the current and projected problems with a centralized CD&R method is the free flight (FF) concept [32]. In this concept, the responsibility of assuring traffic separation is transitioned from ATC to the individual aircraft (airborne separation) [32]. All aircraft are allowed the fly their preferred route. Furthermore, in the latest Federal Aviation Administration (FAA) ConOps, it was stated that a Detect and Avoid (DAA) mechanisms for Beyond Visual Line of Sight (BVLOS) operations of a UAV is required [60].

CD&R thus plays a vital role in the current developments in the fields of ATM and UTM. This research focuses on a class of conflict resolution (CR) algorithms which are dependent on the concept of Velocity Obstacles (VO) [21] for obstacle avoidance. An example of an algorithm is the conflict resolution method based on the Solution Space Diagram (SSD) [2]. In a conflict with two aircraft, the SSD method can achieve a geometrically optimal solution [16], but in conflicts with multiple aircraft, the solution can become sub-optimal. The sub-optimal solution is caused by the complex dynamics which arise when multiple aircraft perform conflict avoidance maneuvers simultaneously. This reduces the set of available avoidance maneuvers which avoid all conflicts. In extreme situations, potentially no maneuver is available that resolves all conflicts.

Analytically establishing a set of rules, which are used to select an avoidance maneuver for any conflict scenario, is extremely hard if not impossible. Deciding on which conflict avoidance maneuver to select based on the current conflict geometry with an approximated model of the system dynamics, is a problem for which Artificial Intelligence (AI) could provide promising solutions. The specific field in AI which focuses on such problems is called Reinforcement Learning (RL). The performance of RL models exceed expert human-level performance on strategic games. For example, a state-of-the-art agent called AlphaGo was able to beat the world champion in Go [85] and more recently, the Alphastar program defeated a top professional player in Starcraft II [95]. This research aims to improve the strategy of an existing CD&R method by leveraging the ability of RL to synthesize strategies.

1.1. Thesis Objective and Research Questions

Shifting the ATC from a centralized approach to a more decentralized approach, which allows an aircraft to determine its own control strategy, facilitates an increase in air traffic volume and a more optimized flight path [32]. Such advantages make research into decentralized CD&R algorithms imperative. Ideally, a decentralized algorithm would allow aircraft to always find the optimal solution, with respect to concepts such as safety and efficiency [87], based on its own observations in any air traffic density and conflict geometry. Unfortunately, such a method does not exist yet. The objective of this research is to improve upon an existing conflict resolution method that implements velocity obstacles and to close the gap between the current limitations of the conflict resolution algorithms and the portrayed perfect algorithm. A more formal definition of the research objective is:

"*Use reinforcement learning to apply conflict prioritization in a multi-aircraft setting in order to improve the efficiency of a velocity obstacle based conflict resolution method at high traffic densities*"

A joint solution indicates that the VOs of all conflicting aircraft are incorporated in the computation of an avoidance maneuver. In velocity obstacle based conflict resolution methods which apply a joint solution [94, 2], the solution space is quickly saturated at high traffic densities. In the extreme this results in no feasible solution. Performing conflict prioritization to increase the available solution space can improve on a joint solution method [68]. In [68], a fixed prioritization rule set was applied to the SSD. With increasing level of priority, the rules were based on lookahead time, time to loss of separation, distance to loss of separation and distance to closest point of approach. The definition of these terms will be provided in section 2.1. The set of rules resulted in an improved CD&R method. It however remains hard to synthesize a fixed ruleset which can be applied to a dynamic multi-aircraft conflict with a continuously changing conflict geometry. Therefore, this research proposes to learn a dynamic strategy for conflict prioritization with RL. The simulations in this research are performed in the BlueSky environment. BlueSky is a simulator which is established to facilitate a general platform on which ATC research can be performed. This allows for better comparison of ATC research [41]. The research objective can be divided into 8 research activities with accompanying research questions.

The research activities are mostly sequential activities which means that the work on research activity 2 starts when work on research activity 1 is finished. Some activities can be performed concurrently or have to be executed in an iterative fashion. The subsequent section on research approach (section 1.2) illustrates the relations between the various activities. Research Activities 1,2,3,4 and 5 will be performed in the preliminary analysis.

Research Activity 1: Selection of conflict resolution method.

The CD&R methods which are considered in this research are the Optimal Reciprocal Collision Avoidance (ORCA) [94] and the SSD method [2]. The original SSD algorithm is considered without the prioritization rules of [68], because the goal of the research is to develop a model which learns its own rules. To decide on the method to which RL will be applied, the following research questions are stated:

a) *Which VO-based conflict resolution methods are suitable for this research?*

b) *How can conflict prioritization be applied to the selected methods?*

c) *What information about the conflict can be extracted from the conflict resolution method which can be used as input for the RL model?*

Research Activity 2: Select a reinforcement learning model which can be used in a high traffic density traffic scenario with multiple controlled aircraft.

The choice for a model structure type is crucial since it will partly determine the success of the final conflict resolution method. This literature review does not aim to be a description of all available RL concepts and algorithms. However, to be able to justify the choice for the RL model and to properly understand the inner workings of the models which is deemed essential in the implementation phase, a theoretical study on the relevant concepts and methods is required. The accompanying research question are:

1. *What are the basic concepts of RL?*
2. *Which categories of methods adhere to the requirements imposed by this research?*
3. *Within the selected categories, which methods are available from comparable research in literature?*
4. *How does the algorithm cope with challenges imposed by multi-agent reinforcement learning?*

Research Activity 3: Define single-agent case for proof-of-concept

In the preliminary analysis, a proof-of-concept for a simple single agent setting is the first hurdle that has to be overcome on the road to design the final model. With the reinforcement learning model that is chosen from literature in Research Activity 1, an experiment is designed to validate that conflict prioritization with the selected RL model is feasible. It should be noted that in the single-agent experiment multiple aircraft are present, but only one is controlled by the RL model. The ensuing research questions can be formulated as:

1. *How to design a simple proof-of-concept experiment which can be extended into a more complex experiment?*
2. *What are the characteristics of the observation space considering that each agent only has partial observability?*
3. *How is the action space specified?*
4. *How can safety be incorporated in the formulation of the reward function?*
5. *What are the hyperparameter settings of the RL model?*

Research Activity 4: Define multi-agent case for proof-of-concept cooperative behaviour between aircraft

The results of the preliminary thesis should be the fundament on which the successor of the preliminary thesis, the actual thesis, is built. The preliminary experiments ought to provide more in-depth knowledge on experimental and modelling principles in the relevant context. The relevant context with regard to this research is the application of reinforcement learning to conflict prioritization in CD&R algorithms which implement VOs in a multi-agent setting. The successive experiment is conducted in a multi-agent setting. The RL model controls conflict prioritization for at least 2 aircraft. The rationale behind the multi-agent experiment is to discover whether aircraft can cooperate to find a solution. Beside similar research questions as in Research activity 3, additional research questions arise:

1. *How will the RL model be implemented in a multi-agent setting?*
2. *Can aircraft efficiently cooperate without sharing intent?*

Research Activity 5: Synthesize proposal for experiment design in the main phase of the thesis.

Based on the results from the single-agent and multi-agent experiment from Research Activity 1 and Research Activity 2 respectively, a training methodology and experiment design for the main part of thesis will be established. The research questions belonging to this objective are:

1. *What are the specifications of the training architecture?*
2. *What is a relevant structure for the traffic scenario in the experiments?*

3. *What are the model settings (observation space, action space, hyperparameters)?*

Research Activity 6,7 and 8 will be performed in the final part of the thesis.

Research Activity 6: Implement and optimize the model settings of the multi-agent scenario.

The multi-agent experiment is initially constructed according to proposal from Research Activity 5. The goal of research activity 6 is twofold: to implement the chosen multi-agent architecture in BlueSky and to optimize performance. The performance of the method is evaluated with Research activity 6 and 7. Therefore, the second goal requires iterative execution of activity 6,7 and 8. The corresponding research questions can be defined:

1. *Which model settings lead to a convergent scenario?*
2. *What are the optimal model settings for the multi-agent experiment?*

Research Activity 7: Evaluate performance in the multi-agent experimental setting of conflict resolution algorithm with conflict prioritization.

The RL model is optimized based on the objectives in the reward function. The RL model is therefore expected to show high performance on the dependent variables which greatly correlate to the reward function components. Nevertheless, it is relevant to investigate the relationship between the model settings (with reward function design) and a predefined set of evaluation parameters. The related research questions can be stated as:

1. *Based on what performance metrics is performance measured?*
2. *How to correctly assess the safety and efficiency of the conflict resolution method?*
3. *Are there recurring phenomena in the behaviour of the aircraft in a multi-aircraft conflict which can be described with fixed-rules?*

Research Activity 8: Compare performance with original conflict resolution method without conflict prioritization.

Finally, the performance of the trained RL model is compared with the original implementation of the SSD method [2]. Both methods are applied to similar scenarios to allow for adequate comparison. The research questions are:

1. *Based on the established evaluation criteria, what are the discrepancies between the two methods?*
2. *What are the differences in aircraft behaviour?*
3. *What are the recommendation for future work?*

1.2. Research Approach

The results from the research activities can be combined to establish a solution to the thesis objective. This section illustrates the relationship between the various research activities. The overview for the preliminary and final phase can be found in Figure 1.1 and Figure 1.2 respectively.

On the left-hand side of Figure 1.1, the single-agent experiment is described (Research Activity 3). Based on the chosen CR method and RL model from literature, the model settings are initialized (Research Activity 1,2). A scenario in BlueSky is defined and the model alternates between training and adjusting of model settings until a successful agent has been trained. A successful agent is a rather ambiguous term, but in this scenario corresponds to the agent successfully maneuvering through the scenario. More details will be explained in chapter 3.

Thereafter, a successful single-agent scenario can be transformed to a multi-agent scenario for the second experiment (Research Activity 4). Achieving a convergent model which showcases desirable properties is expected to be significantly more difficult compared to the single-agent case due to the challenges of multi-agent reinforcement learning described in subsection 2.3.2. The knowledge gained from the single-agent case is utilized in the formation of the model setup for the multi-agent case.

Finally, the results of the single- and multi-agent experiments are combined to synthesize a proposal for the experiment design in the subsequent thesis phase (Research Activity 5). The term proposal is employed since the design is based on current knowledge. In the implementation phase, deviations from the proposal can occur in further design iterations.

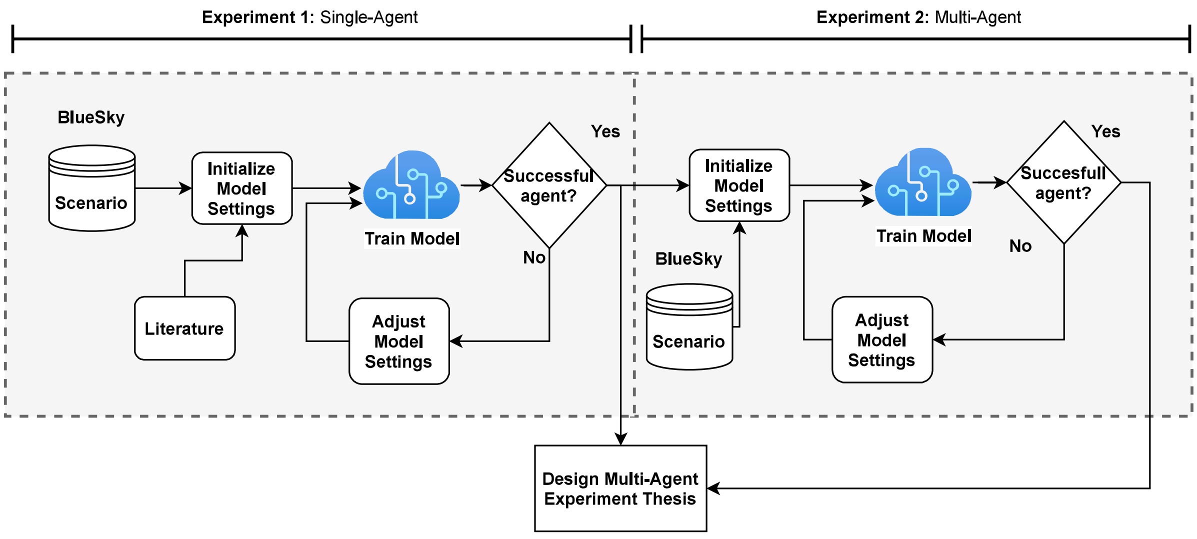


Figure 1.1: Schematic overview of single- and multi-agent experiment and their relation to each other and the design of the multi-agent experiment in the main part of the thesis

The approach in the final thesis is similar to the preliminary phase. An overview of the methodology is provided in Figure 1.2. The starting point is implementing the proposal from the preliminary phase (Research Activity 6). After that, the model is trained and the performance is evaluated (Research Activity 7) and compared to the original CR method (Research Activity 8). The model settings are updated until satisfactory results are attained.

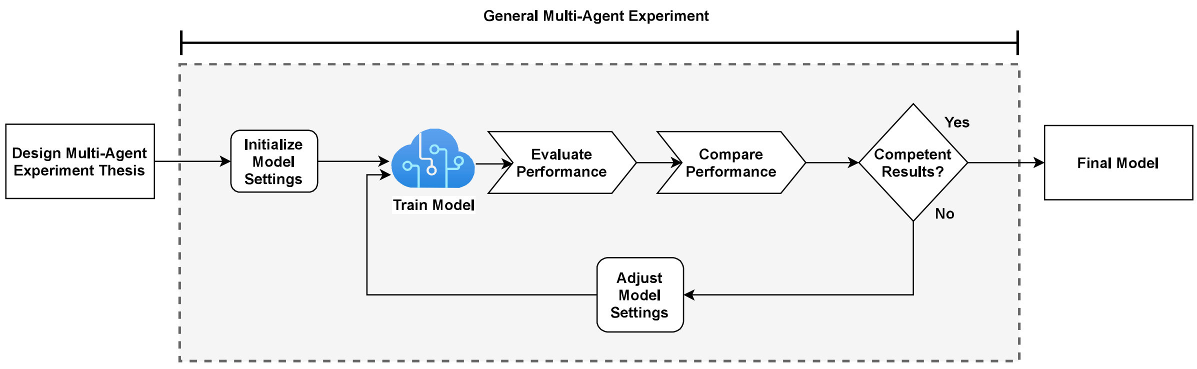


Figure 1.2: Schematic overview of general multi-agent experiment and its relation to the final model

1.3. Research Scope

To narrow the scope of this research, a list of general assumptions is provided in this section. If deemed necessary, more assumptions are stated throughout the research to further narrow the scope. The following general assumptions apply to this research:

- **No wind and turbulence in experiments**. The experiments are performed in the BlueSky simulator [41] in which wind and turbulence are deactivated. In the experiments perfect weather conditions are assumed with no wind or turbulence. In such perfect conditions the True Air Speed (TAS) is equal to the ground speed (GS).
- **Homogeneity of CD&R methods for aircraft**. It is assumed that all aircraft which have the CD&R method activated, implement the same algorithm.
- **Homogeneity in aircraft model**. When conducting an experiment, the aircraft all have the same model type which translates to similar constraints that are imposed on the agents in terms of performance.
- **No restrictions with regard to air space**. The aircraft are allowed to freely maneuver through the air space and follow any flight path that they desire during the entirety of every experiment. This assumption adheres to the ideas of the free-flight concept [31].
- **Aircraft instantaneous obtain an unbiased measurement of the states of surrounding aircraft**. In real-life, the Automatic Dependent Surveillance Broadcast (ADS-B) provides a noisy estimate of the aircraft state with a delay in time.

1.4. Outline Report Structure

Section 2.1 elaborates on the various facets of conflict detection and resolution methods with a focus on methods that implement velocity obstacles. Subsequently, the concepts of reinforcement learning in a single- and multi-agent setting are elaborated upon. Thereafter, chapter 3 explains the experiments performed in the preliminary analysis based on knowledge gained from chapter 2. Finally, chapter 2 also provides a proposal on the research approach in the final thesis.

2
Literature Review

In this chapter a description of the literature required to answer the research questions is provided. Section 2.1 elaborates on conflict detection and resolution for manned and unmanned aviation. Thereafter, the basic principles of reinforcement learning are described in section 2.2. Finally, the advanced reinforcement learning concepts are described in section 2.3.

2.1. Conflict Detection and Resolution

The basic concepts and definitions in the field of CD&R are described in subsection 2.1.1. A taxonomy of CD&R methods for manned and unmanned aviation is used to define the scope of this research. Subsequently, the taxonomy is used to provide a perspective on how this research relates to the spectrum of CD&R methods in subsection 2.1.2. Section 2.1.3 further elaborates on the concept of conflict detection. The principles of a velocity obstacles are elaborated upon in subsection 2.1.3. Thereafter, the Solution Space Diagram (SSD) and the Optimal Reciprocal Collision Avoidance (ORCA) method are discussed in subsection 2.1.5 and subsection 2.1.6 respectively. The performance metrics which can be used to evaluate the CD&R methods are explained in subsection 2.1.8.

2.1.1. Basic Concepts and Definitions

The concepts and definitions explained in this section are required as basic knowledge for the remainder of section 2.1. Part of the terms introduced in subsection 2.1.1 will be more elaborately discussed in subsequent sections, often from a more mathematical perspective. The assumptions stated in subsection 2.1.1 are further justified in subsection 2.1.2

The first concept concerns the separation limits in manned and unmanned aviation. The horizontal and vertical separation minimum separation requirements for manned aircraft are defined by the ICAO [38]. The horizontal requirement is 5 nm when radar, ADS-B or MLAT is utilised to determine the aircraft position. The vertical minimum distance is 1000 ft below FL290. Exceptions to these limits do exist in specific situations. The separation distance has to be obeyed with respect to all heading angles thus forming a circle with the aircraft at its center. This circle is called the protective zone. The protective zone with horizontal separation limit R_p and vertical separation limit h_p are illustrated by Figure 2.2.

Regulation for unmanned aircraft has not been properly established yet and is still in development. Therefore, exact numbers on the horizontal and vertical separation limits do not exist. The Bubbles project aims to solve this problem and develop a framework for separation limits [81]. In research on conflict detection and resolution for UAVs, the horizontal minimum separation distance was selected between 15 m and 400 m [29, 100, 72]. This range can be employed as a reference.

This research will focus on the conflict detection and resolution in the horizontal plane. Therefore, the term minimum separation distance will indicate the horizontal minimum separation distance in future sections.

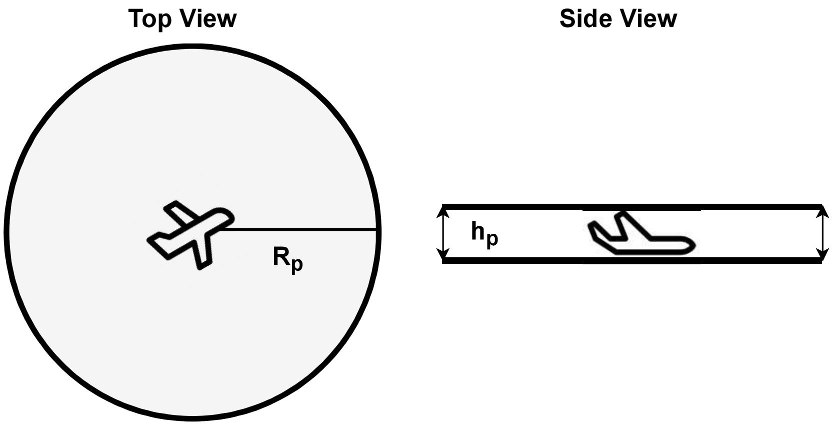


Figure 2.1: Schematic Overview of horizontal (R_P) and vertical separation limits (h_p)

It is assumed that the aircraft propagate their current state in a linear manner [73]. A loss of separation (LoS) for ownship occurs when an intruder penetrates the protected zone. The terminology ownship and intruder is adopted from [73] and denotes the perspective of the conflict situation. A conflict occurs when a LoS is predicted based on linear propagation of the current states if the involved aircraft. In Figure 2.2 a conflict takes place at time t_1. If no adaption is made to the projected flight path of both aircraft, a LoS between ownship and intruder occurs at time t_2

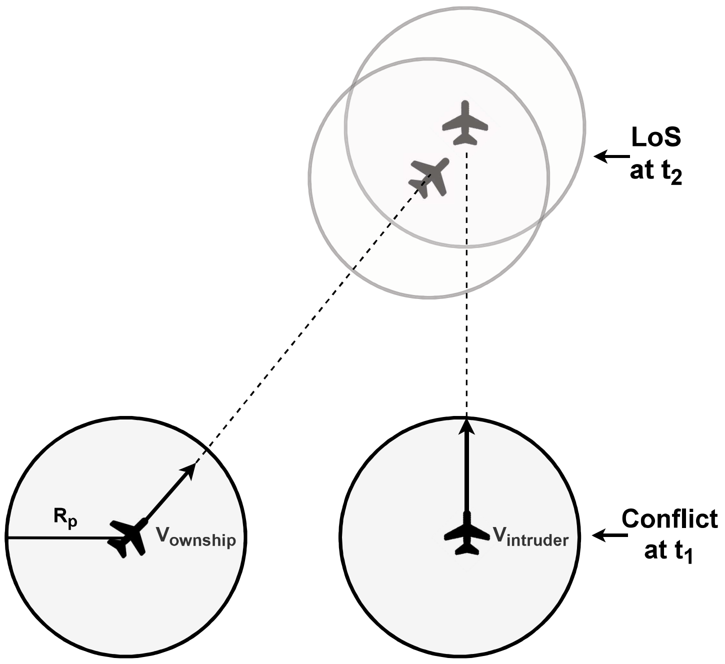


Figure 2.2: Scenario in which a conflict between ownship and intruder is present at t_1 and a loss of separation (LoS) at t_2. Ownship has velocity $V_{ownship}$ and radius of protected zone R_p.

A summary of the terms introduced in subsection 2.1.1, along with definitions for conflict detection and conflict resolution that are used in this research, can be found in Table 2.1.

Table 2.1: Summary basic definitions in field of Conflict Detection and Resolution (CDR)

Term	**Definition**
Protected Zone	Zone around aircraft which is generated by vertical and horizontal separation limits that have to be obeyed for all heading angles
Conflict	A conflict arises when the propagated trajectory of an intruder results in a conflict with ownship within a pre-specified time interval into the future
Loss of Separation	A LoS denotes the penetration of the protected zone of ownship by an intruder
Conflict Detection	Conflict detection describes the mechanism which monitors whether conflicts are present
Conflict Resolution	Conflict resolution is applied to establish a conflict-free trajectory once a conflict is detected [25].

2.1.2. Conflict Detection and Resolution Drones Taxonomy

This research aims to improve on an existing CD&R method which can be applied to both manned and unmanned aviation. Within the landscape of CD&R, a great abundance of methods exist. To provide context on the various characteristics of CD&R algorithms and the relation between the scope of this research and the total spectrum of algorithms, a taxonomy of CD&R algorithms is provided in this section. The assumptions stated in section 1.3 are utilized to define the subspace in which the to be researched algorithms operate.

The taxonomy from [73] is utilized to provide a structured overview of the available CD&R methods for manned and unmanned aircraft, which aims to extend the taxonomy of Kuchar and Yang [47] by incorporating more recent CD&R concepts. An overview of the various categories of the taxonomy used to define each CD&R method can be found in Figure 2.3. The highlighted boxes in Figure 2.3 indicate the characteristics of the relevant algorithms for this research. A motivation on the scope is provided in the remainder of this section.

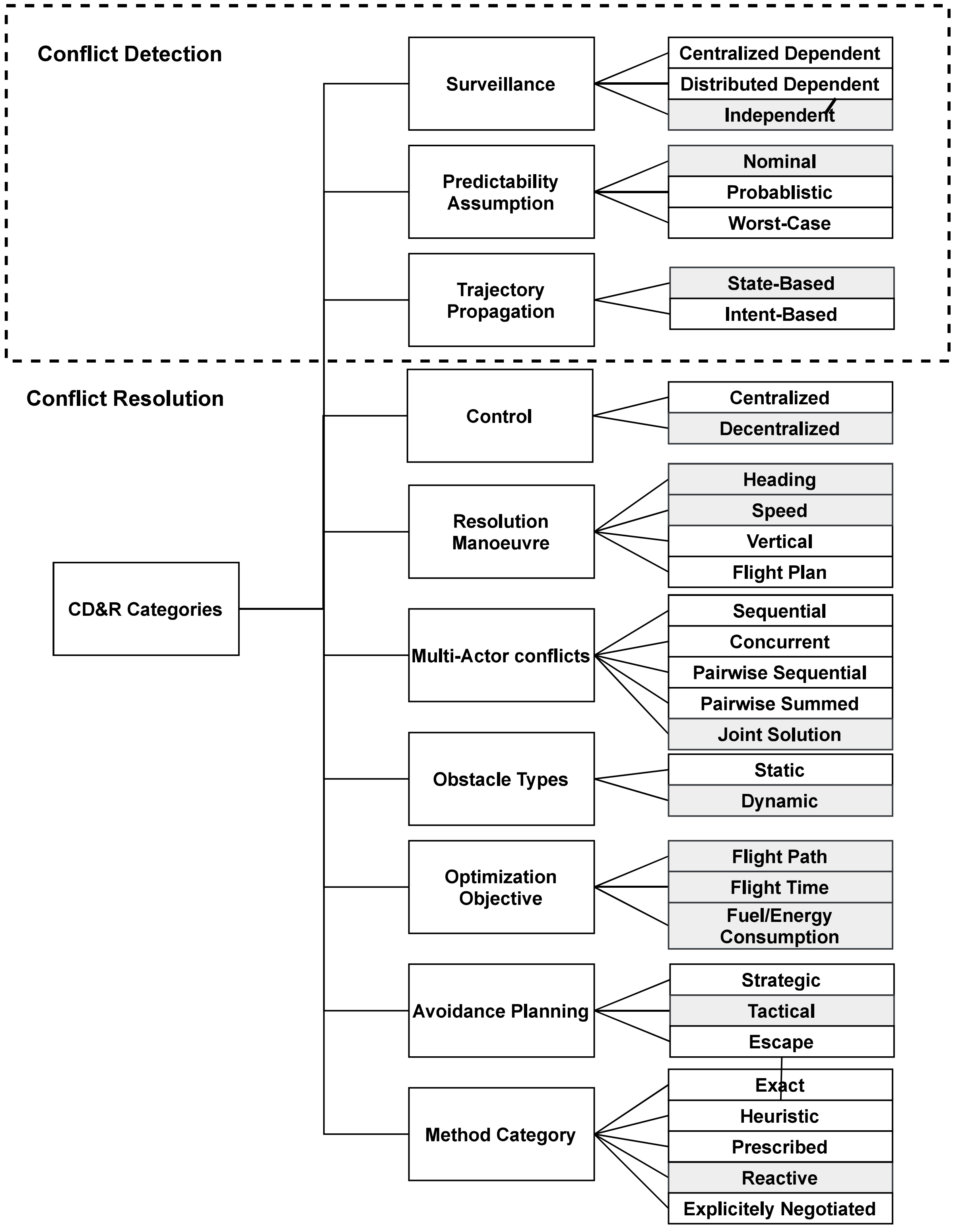
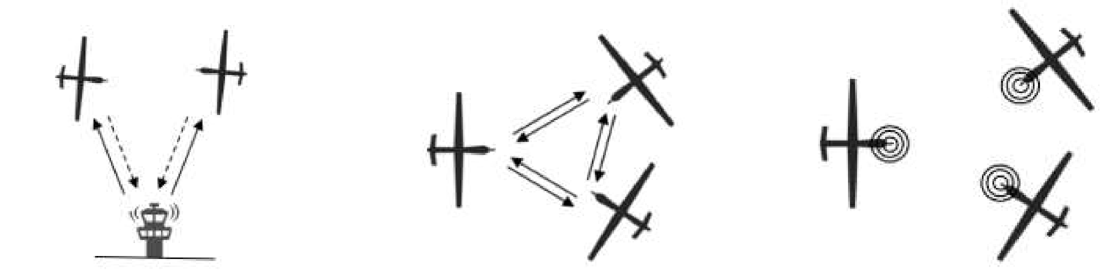


Figure 2.3: Overview of categorization CD&R algorithms [73]

Types of surveillance
Aircraft surveillance type denotes whether the aircraft is able to autonomously gather data relevant for CDR or if the aircraft is dependent on incoming data from external systems. Three main surveillance types exist

namely centralized dependent, distributed dependent and independent surveillance or DAA. In a centralized dependent system, information is retrieved from one general station. In a decentralized dependent system the aircraft broadcast information to each other. In manned aviation, a system called Automatic Dependent Surveillance-Broadcast (ADS-B) is in place which can transmit state information on the aircraft position, ground speed and vertical rate which can be used by other aircraft or ATC [86]. Finally, in this research it is assumed that UAVs have an independent system in which static and dynamic obstacles are detected with sensors integrated in the drone. The various types of surveillance are illustrated with Figure 2.4.



(a) Centralised dependent surveillance **(b)** Distributed dependent surveillance **(c)** Independent Surveillance

Figure 2.4: Overview of surveillance types from [73]

It is assumed that the aircraft is capable to detect the relative distance and velocity to all obstacles that are considered to be an intruder in a 2-dimensional plane.

Control
In manned aviation, ATC is responsible for the separation of all aircraft within their monitored airspace. This is a centralized control approach since ATC provides a solution to all aircraft from a central point, the ATC tower. In a distributed approach the computation of the solution is divided over each individual aircraft thus reducing the computational limitations to which centralized solutions have to adhere to. Since the bottleneck in ATC is the workload of the ATCos even with a limited number of aircraft [27], with the expected increase in the number of drones [14], a distributed solution becomes a very viable and attractive option. Another disadvantage about a centralized approach is the need for communication. In the case of manned aviation, the aircraft need to send information about their state to the ATC tower which computes a global solution. This global solution then needs to be communicated to all aircraft. The delay caused by communication limits the speed at which aircraft can adapt their state and no central coordination is possible in case of communication failure. In a distributed system, the lack of global coordination could incite hazardous situations if the aircraft involved in the conflict do not possess sufficient information on the intent of other aircraft to form a globally optimal solution [71].

The current approach in manned aviation is a centralized approach which is expected to experience problems with an increase in air traffic. For UAVs no centralized system is in place. Therefore, this research focuses on distributed control methods in which one of the challenges will be to cope with the lack of global coordination.

Trajectory Propagation
The possible future trajectory of an aircraft can be constructed based on current state (i.e. state-based) or with the inclusion of future intent (i.e. intent-based). A state-based method assumes linear propagation of the current state while an intent-based method can incorporate variations in heading and speed based on readily available flight plan.

Sharing of intent entails data transmission through communication between the aircraft. Various problems concerning data transmission were discussed in the section on control hence the trajectory propagation of the CD&R method in this research assumes state-based trajectory propagation.

Predictability Assumption
In the process of predicting the future states and thus trajectory of an aircraft including assumptions about disturbances (irregular wind gusts, uncoordinated traffic) can be made on three levels: nominal, worst-case

and probabilistic. The nominal assumption assumes no disturbances. The worst-case assumption considers all events that could potentially influence the trajectory. With the probabilistic assumption, a probability is assigned to each of the possible future paths and the decision on a conflict maneuver is dependent on the most probable path.

The more information on disturbance factors is included, the higher the level of accuracy a conflict resolution maneuver can obtain. The increased accuracy of the solution comes with a higher computational cost and therefore the nominal predictability assumption is selected.

Avoidance Planning
Avoidance planning can occur on three levels. Strategic planning of conflict resolution maneuvers significantly alters the original flight path in an attempt to avoid unnecessary conflicts. Tactical maneuvers are in the mid-range and try to keep the minimum separation requirements intact while also minimizing the deviation from the original flight path. As a last resort, there are the escape maneuvers which do not care about flight path optimisation and only want to avoid a collision. The various types of avoidance planning maneuvers are visualized in Figure 2.5. Typical ranges for look-ahead time that belong to the different types of avoidance planning can be found in Table 2.2 [73]. In manned aviation ATC and CD&R methods take care of strategic and tactical avoidance maneuvers. For escape maneuvers, a Traffic Alert and Collision Avoidance System (TCAS) can compute a pairwise solution for a two-aircraft conflict [48]. For UAVs all strategic, tactical and escape avoidance maneuvers depend on the CD&R algorithm. Due to the limitations of the on-board sensing and processing systems, the avoidance maneuvers of UAVs can produce tactical and escape maneuvers only. Consequently, tactical avoidance planning type is assumed.

Table 2.2: Different types of avoidance planning with accompanying look-ahead time ranges

Avoidance Planning Type	Look-ahead Time Range	Unit
Strategic	>20	min
Tactical	3-20	min
Escape	< 3	min

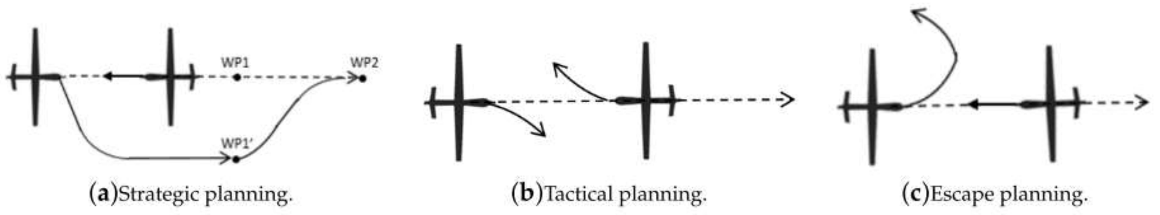


Figure 2.5: Overview of avoidance planning types from [73]

Maneuver Employed for Resolution
An aircraft can resort to several options when choosing a resolution manoeuvre. The avoidance manoeuvre can translate into a change in current heading, velocity, altitude and/or a change in flight plan. Even though three-dimensional conflict resolution methods which implement velocity obstacles exist [12], it is decided to limit the scope to methods which employ heading and velocity changes to limit the computational complexity. Besides, it was found that horizontal maneuvers are preferred for both passenger comfort and fuel consumption [10].

Approach to Multi-Actor Conflicts
Multi-actor conflicts are conflicts which include more than two aircraft. Such conflicts can be solved in a sequential manner in which a prioritization is made between the various aircraft involved in the conflict [35]. The second method is to solve the conflicts in a concurrent manner where all new trajectories for all aircraft are computed at once [65]. Solutions in a decentralized approach can be divided between joint solutions, pairwise sequential solutions and pairwise summed solutions. In a joint solution, all other aircraft that are

considered intruders based on user defined criteria (e.g. relative distance), are taken into account when computing a solution that solves all conflicts concurrently. In a pairwise sequential solution, conflicts pairs are solved individually based on the conflict prioritization order. Finally, in a pairwise summed solution the avoidance maneuvers of the individual pairs are added together into one conflict avoidance maneuver.

In chapter 1, it was already explained that this research focuses on joint solution methods.

Obstacle types
A CD&R method can be designed to cope with static obstacles, dynamic obstacles or a combination of both. It is no guarantee that a CD&R method which is designed for dynamic obstacles can automatically avoid static obstacles. Static obstacles are neglected in this research.

Optimisation Objective
The performance of a CD&R method can be measured based on several components. Safety is naturally the most obvious and important measure. Besides safety, it can be important to choose the resolution maneuver that alleviates the consequences on the fuel consumption or flight time.

The optimisation objectives partly dictate the reward structure of the reinforcement learning algorithm. Therefore, at this point it is too early to fix the optimisation objectives. However, this research starts with an already existing CD&R algorithm. Similar dependent variables that were used to assess the original algorithm can thus be used. In this manner, the discrepancies between the original method and the algorithm of this research can be investigated.

Method Categories
With a distributed control scheme, the conflict resolution categories are divided based on the type of avoidance manoeuvres. The categories are prescribed, reactive and explicitly negotiated. In prescribed methods all aircraft involved act according to a set of predefined rules, while in reactive methods the aircraft compute their avoidance manoeuvres from the same conflict resolution algorithm that takes conflict geometry as input. Finally, in explicitly negotiated all aircraft involved in the conflict communicate with each other such that all aircraft are aware of the intended conflict avoidance strategy of the other aircraft.

The VO-based conflict resolution methods decide on the avoidance maneuver based on the conflict geometry. Therefore, the method of this research falls in the reactive category.

Concluding Remarks
The initial scope of the research is defined in subsection 2.1.2. This does not mean that the scope is definite and cannot be altered any more. During this research, it could be concluded that the current scope is not adequately defined and needs to be changed. It will be clearly stated when assumptions are added/removed. The algorithms that fit the criteria of the scope and are selected for further research are the SSD [2] and Optimal Reciprocal Collision Avoidance [94] which are discussed in subsection 2.1.5 and subsection 2.1.6.

2.1.3. Conflict Detection

An introduction to conflict detection was given in subsection 2.1.1. This section explains further elaborates on the characteristics of a conflict situation. The Closest Point of Approach (CPA) is the point at which minimum separation between the ownship and intruder aircraft is reached. The distance to Closest Point of Approach (d_{cpa}) is defined as the distance between intruder and the closest point of approach. The Time to Closest Point of Approach (t_{cpa}) is the time until the closest point of approach is reached. A conflict is detected when d_{cpa} is smaller than the radius of the protected zone R_p.

To derive the equations for t_{cpa} and d_{cpa}, start with a scenario in which the trajectories of two aircraft are derived from their current state in a linear manner. The distance between the two aircraft starting at time t with their initial position at $t = 0$ is equal to Equation 2.1 in which d_{ij} denotes the distance between aircraft i and j in m, $\mathbf{x}$ the position vector with unit m, v the velocity vector with unit m/s, the vector $\mathbf{d}_{rel}$ represents the relative position between the two aircraft in m and $\mathbf{v}_{rel}$ denotes the relative velocity vector between the two aircraft with unit m/s [37].

$$\begin{aligned} d_{ij}(t) &= |\mathbf{x}_i(t) - \mathbf{x}_j(t)| \\ &= |\mathbf{x}_i(0) - \mathbf{x}_j(0) + (\mathbf{v}_i - \mathbf{v}_j) \cdot t| \\ &= |\mathbf{d}_{rel}(t) + \mathbf{v}_{rel}(t) \cdot t| \end{aligned} \tag{2.1}$$

The goal is to find time t which minimizes the distance d_{ij}. This can be accomplished by setting the derivative of $d_{ij}(t)$ equal to 0. To deal with the negative values the derivative of d_{ij}^2 is computed. The result of this derivation are the equations for t_{cpa} and d_{cpa} which can be found in Equation 2.2 and Equation 2.3 respectively.

$$t_{cpa} = -\frac{\mathbf{d}_{rel} \cdot \mathbf{v}_{rel}}{|\mathbf{v}_{rel}|^2} \tag{2.2}$$

$$d_{CPA} = \sqrt{\mathbf{d}_{rel}^2 - t_{CPA}^2 \cdot \mathbf{v}_{rel}^2} \tag{2.3}$$

With aid of geometric relations, Equation 2.4 can be derived in which t_{in} represents the time until a loss of separation occurs and t_{out} when the conflict is resolved. A conflict occurs if the current path of an aircraft will result in a loss of separation within a pre-defined look-ahead time window τ. Combined with the stated assumptions on the CD&R methods in this research, this translates to $t_{in} \leq \tau$ and $d_{cpa} < R_{PZ}$ [73]. The term t_{in} can also be referred to as the time to loss of separation t_{LoS}.

$$t_{in}, t_{out} = t_{CPA} \pm \frac{\sqrt{R_{PZ}^2 - d_{CPA}^2}}{\mathbf{v}_{rel}} \tag{2.4}$$

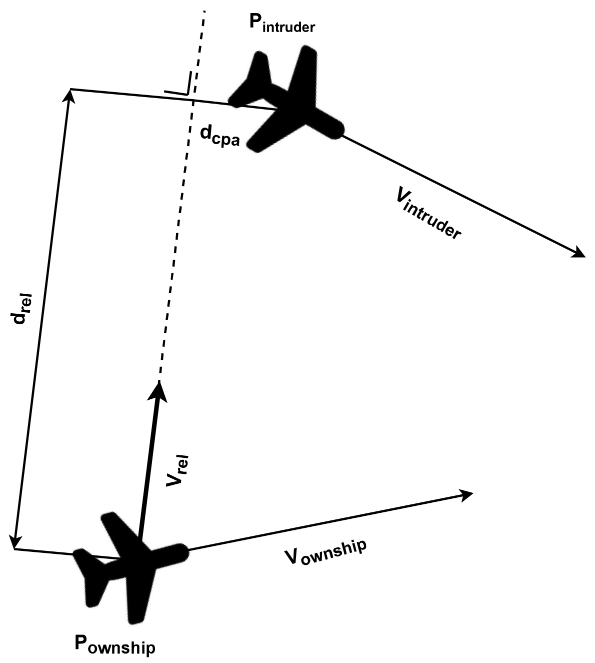


Figure 2.6: Description of relation between $d_{cpa}, d_{rel}, v_{rel}$ and positions and velocities of ownship and intruder, adapted from [73]

2.1.4. Velocity Obstacles

This section first describes the original paper which implemented velocity obstacles. Thereafter, it is described how the velocity obstacles can be applied in the field of aviation.

Original Velocity Obstacles

The idea of implementing velocity obstacles for 2-dimensional motion planning and obstacle avoidance was already introduced in the previous century [21]. The collision cone described in the paper contains a set of all relative velocities between robots A and B which result in collision at some time in the future. The construction process of a collision cone can be explained with aid of Figure 2.7 and Figure 2.8. In Figure 2.7 the initial situation can be found at t_0 in which two circular shaped robots A and B with radii r_A and r_B have velocities v_A and v_B. In Figure 2.8 r_a is subtracted from robot A and added to r_b forming circle $\hat{B}$, $\mathbf{v}_{A,B}$ is the relative velocity vector between A and B and $\lambda_{A,B}$ is a line which extends $\mathbf{v}_{A,B}$ to infinity, λ_r and λ_f are tangent lines to $\hat{B}$ from $\hat{A}$.

Intuitively, if $\lambda_{A,B}$ intersects with circle $\hat{B}$, this means that at some time t in the future, a collision will occur between robot A and B with the current relative velocity $\mathbf{v}_{A,B}$. A more formal and mathematical definition of the collision cone $CC_{A,B}$ can be found in Equation 2.5 which states that any relative velocity $\mathbf{V}_{A,B}$ for which $\lambda_{A,B}$ intersects with $\hat{B}$ belongs to the set of velocities that compose $CC_{A,B}$.

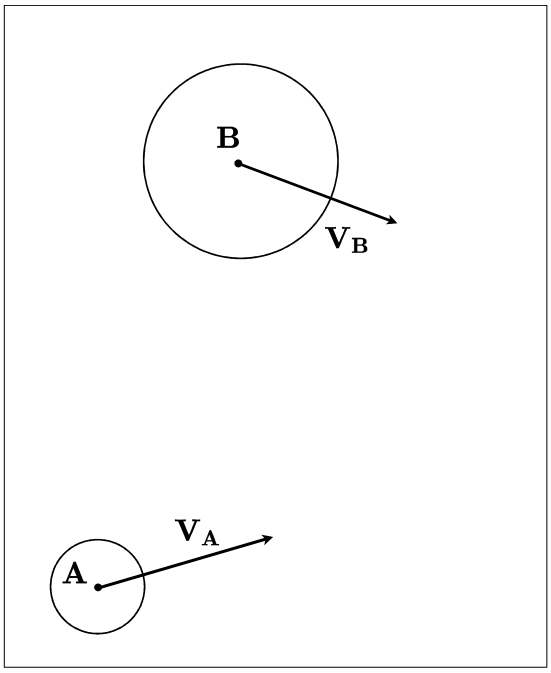


Figure 2.7: Robots A and B are moving with velocities v_A and v_B respectively, adapted from [21]

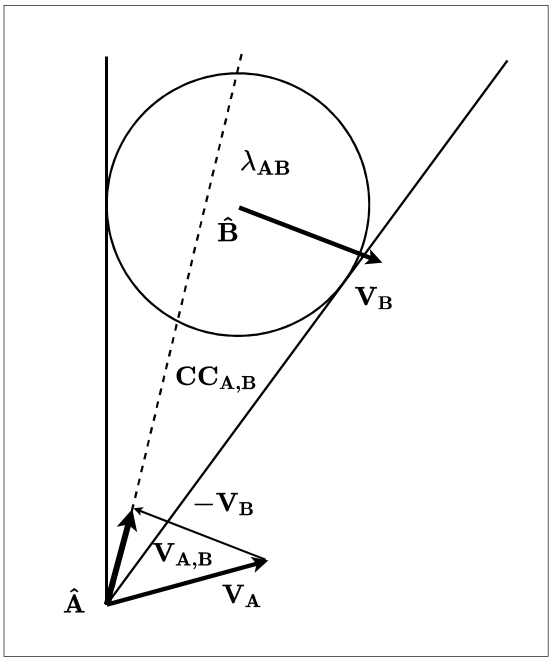


Figure 2.8: The collision cone $CC_{A,B}$ adapted from [21]

$$CC_{A,B} = \{\mathbf{v_{A,B}} | \lambda_{A,B} \cap \hat{B} \neq 0\} \tag{2.5}$$

Any relative velocity outside of $CC_{A,B}$ guarantees a collision-free path for robot A and B [21]. Subsequently, the collision cone can be transformed from the relative velocity plane to the absolute velocity plane. This simplifies the conflict resolution process since avoidance maneuvers can be planned and executed in the absolute velocity plane. Also, this allows the inclusion of transformed collision cones of multiple dynamic and/or static obstacles in the process of selecting a desired velocity vector. This operation can be performed according to Equation 2.6 in which $\oplus$ is the Minkowsky vector sum operator. The collision cone is now renamed as a velocity obstacle.

$$VO = CC_{A,B} \oplus \mathbf{V_B} \tag{2.6}$$

Velocity Obstacles with Aircraft

The construction of velocity obstacles for aerial vehicles can follow a similar methodology as the circular shaped robots, because the aircraft have to obey the minimum horizontal separation distance. The aircraft thus also have a circular shape and the underlying mathematical problem is therefore comparable. If the relative velocity between aircraft A and B is within $CC_{A,B}$, this will results in a loss of separation. The computation of the velocity obstacle for two aircraft is displayed in Figure 2.9 which shows great similarity to the example in section subsection 2.1.3. The shift of the collision cone by the Minkowsky vector sum operator is clearly visualized by Figure 2.9b and Figure 2.9c.

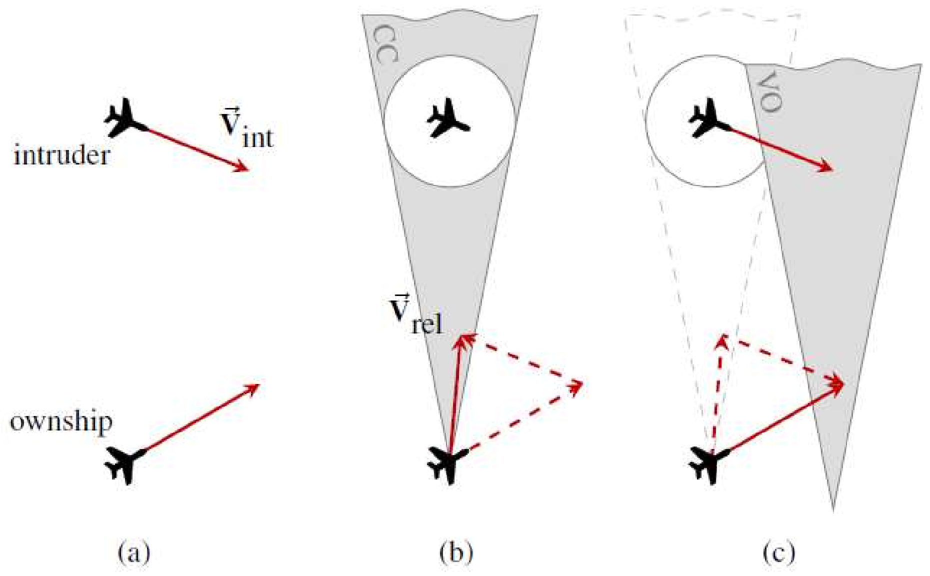


Figure 2.9: The initial situation (a) for which the collision cone for ownship and intruder is constructed (b) and transformed from the relative velocity to the absolute velocity reference plane (c) from [2]

2.1.5. Solution Space Diagram

The first concept similar to the SSD as implemented by [2], was introduced by [93] as the "state vector envelope". The goal of the state vector envelope was to assist the pilot in the aircraft separation assurance task. Subsequent research focused on the assessment of the ATCo workload [26]. High correlation between properties of an SSD and the difficulty of the aircraft merging task as reported by the ATCo were found.

Thereafter, instead of only measuring workload of the ATCo with the SSD, the idea arose to employ the SSD as a tool for the ATCo's to manage air traffic [56]. Significant effects on the reduction of the workload of the ATCo were found by implementing the SSD. Furthermore, experiments were performed to test the acceptance of the SSD as a CD&R tool [5] by ATCo's. Finally and most importantly for this research, the SSD was implemented as a conflict resolution [2].

This section first elaborates on the construction of the SSD. Thereafter, the components which are required to synthesize the SSD and its solutions are described. At last, the SSD as a conflict resolution algorithm is explained.

Construction of Solution Space Diagram

This research implements the SSD in a similar fashion as in [2]. Therefore, the construction process will be similar and is explained with Figure 2.10. Figure 2.10a displays 3 VOs which belong to 3 obstacles. If the velocity vector of ownship is located within one of the VOs, the ownship is in a conflict. Therefore, the union of the 3 VOs is called the set of Forbidden Reachable Velocities (Equation 2.7). In Figure 2.10b the set of reachable velocities (RV) is displayed and is based on the performance limits of the aircraft. The inner circle has radius v_{min} and the outer circle has radius v_{max}. Assuming that no other regulatory constraints are imposed, an aircraft is obviously able to turn to any desired heading hence the circular shape (for every heading angle, the aircraft at least flies at v_{min} and maximally at v_{max}).

$$\text{FV} = \bigcup_{i=1}^{N} \text{VO}_i \tag{2.7}$$

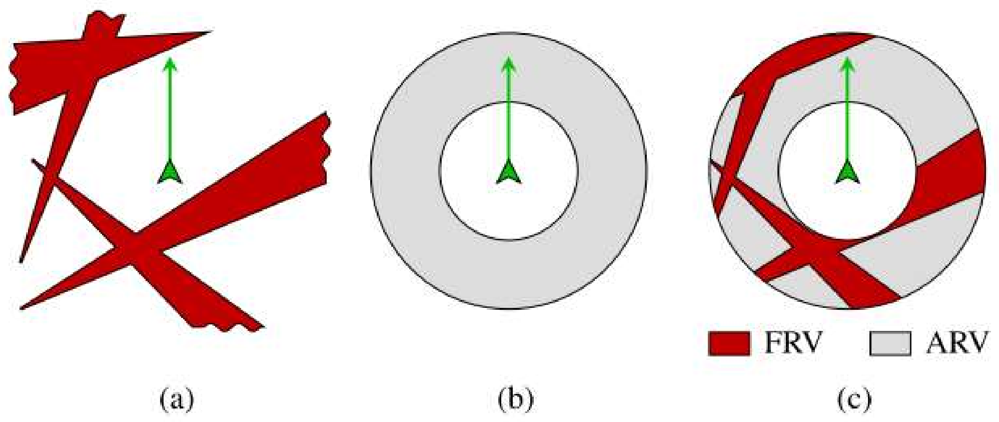


Figure 2.10: Velocity obstacles from 3 intruders (a) are combined with the performance limits of the aircraft (b) to form the SSD (c) [2]

Division of the RV into the Allowed Reachable Velocities (ARV) and Forbidden Reachable Velocities is the final step to synthesize the SSD. The FRV and ARV can be found with Equation 2.8 and Equation 2.9 respectively. Figure 2.10c displays the final result.

$$\mathrm{FRV} = \mathrm{RV} \cap \mathrm{FV} \tag{2.8}$$

$$\mathrm{ARV} = \mathrm{RV} \cap \mathrm{FV}^{C} \tag{2.9}$$

Components of Solution Space Diagram

Selecting the right for the formation of the state space in the reinforcement learning algorithm is a difficult task. Utilizing the entire SSD image as input image has several drawbacks. An image is essentially a 3-D array in which two dimensions represent the length and width of the image measured in pixels. The third dimension denotes the color of each pixel. A variety of ways exist to represent the color of the pixel such as RGB (red-green-blue) or grayscale. However, the image of the SSD will probably need to be compressed in order to train the model in a feasible amount of time. During compression, information from the conflict geometry is lost which is undesired. Furthermore, if VOs from various conflicts are overlapping, the model cannot distinguish individual VOs and has no correct interpretation on the conflict geometry. It is therefore decided that the parameters which are used to construct the SSD will be directly inputted to the model.

A summary of parameters which are utilized in the computational process of the SSD and its solutions can be found in Table 2.3.

Table 2.3: Summary of parameters SSD. Parameters are assumed to belong to ownship if not explicitely defined otherwise.

Parameter	**Unit**
Velocity	m/s
Maximum Velocity	m/s
Minimum Velocity	m/s
Heading	deg
Conflict Angle	deg
Latitude and Longitude intruders	deg
Relative heading intruders	deg
Relative distance intruders	m
Velocity intruders	m/s
Target Heading	deg
Number of conflicting aircraft	[-]
t_{cpa}	s
d_{cpa}	s
t_{los}	s

Solution Space Diagram in Conflict Resolution
The SSD diagram provides a set of allowable reachable velocities, but that is only part of the solution. The second part is to select which velocity vector to choose from the set of available reachable velocities. In Table 2.4, various rules for selecting a velocity vector from the ARV can be found [2]. For conflicts that involve two aircraft, the first rule OPT is geometrically optimal according to Ellerbroek [16]. The other rules also include the ROW rules and a prioritization of VOs.

Table 2.4: Eight rulesets which can be applied to the Solution Space Diagram to compute the velocity vector of an avoidance maneuver adapted from [2]

ID	Priority	Description
1. OPT	✗	Resolve by taking the shortest way out
2. RIGHT	✗	Resolve by only turning right
3. HDG	✗	Resolve by only changing speed
4. SPD	✗	Resolve towards the target heading
5. DEST	✗	Resolve by adhering to the rules of air
6. ROTA	✓	Resolve sequentially while adhering to OPT
7. OPT+	✓	Resolve sequentially while adhering to OPT
8. DEST+	✓	Resolve sequentially while adhering to DEST

The rules can be best explained with an example SSD. The SSD that illustrates the rules without priority and with priority can be found in Figure 2.11 and Figure 2.12 respectively. The numbered yellow points in Figure 2.11 and Figure 2.12 represent the the velocity vector from the ARV that belongs to the solution of the rule. In Figure 2.12, only VOs from which the intruder is within 40 NM are taken into account when computing the resolution velocity vector. For example, in Figure 2.12, the solution for OPT without priority (1) is clearly different than the solution for OPT+ with priority (7).

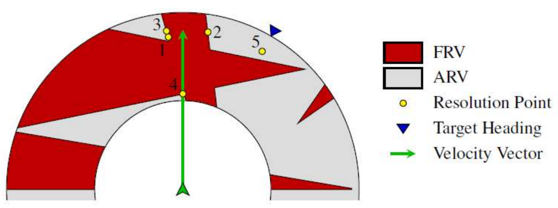


Figure 2.11: Solution points with basic coordination ruleset without priority from [2]

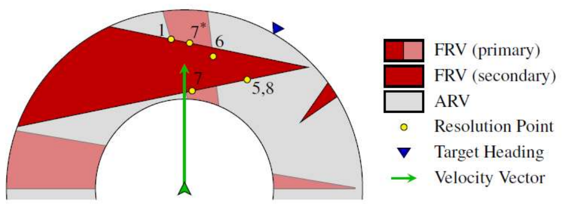


Figure 2.12: Solution points with basic coordination ruleset with with priority from [2]

The shortest path out ruleset showed the most promising results in terms of safety, stability and efficiency. An interesting recommendation from [2] was to take the dynamic behaviour of the SSD into account when constructing the resolution velocity vector. The dynamic behaviour can be included by constructing an SSD in which the states of all aircraft involved are linearly propagated.

2.1.6. Optimal Reciprocal Collision Avoidance Method
Another conflict resolution approach based on VOs is the Optimal Reciprocal Collision Avoidance (ORCA) algorithm. The ORCA method was originally applied to circular robots [94]. It is a decentralized method in which each robot selects a preferred velocity such that all robots are collision free for at least τ seconds. The definition of the VOs is slightly different than in subsection 2.1.3 and is explained with Figure 2.13.

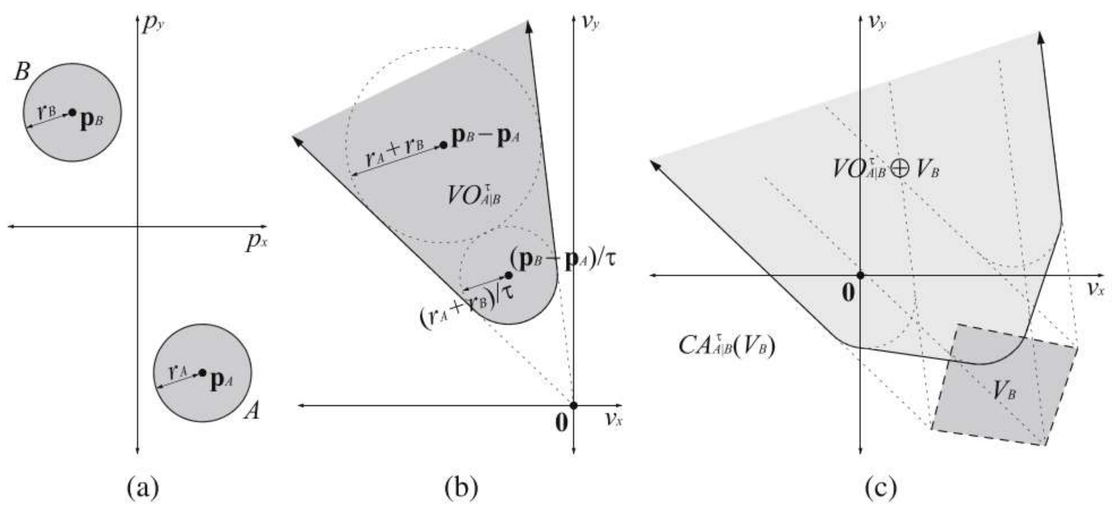


Figure 2.13: The initial situation (a) for which the velocity obstacle $VO^{\tau}_{A|B}$ is constructed (b) which is transformed to the absolute velocity plane assuming robot B selects its velocity from V_B (c), adapted from [94]

The initial situation is similar (Figure 2.13a), but the VO obstacle $VO^{\tau}_{A|B}$ is now determined based on τ seconds in the future causing r_a of robot A to be transformed to $\frac{\mathbf{p}_B - \mathbf{p}_A}{\tau}$. Similarity of the two methods can simply be proven with Equation 2.10. If you take the limit of τ to infinity the truncated cone of Figure 2.13b will approximate the same triangular shape as in subsection 2.1.3. The truncated cone means that the relative velocity between the two robots can be so small that it will not result in a loss of separation within τ seconds.

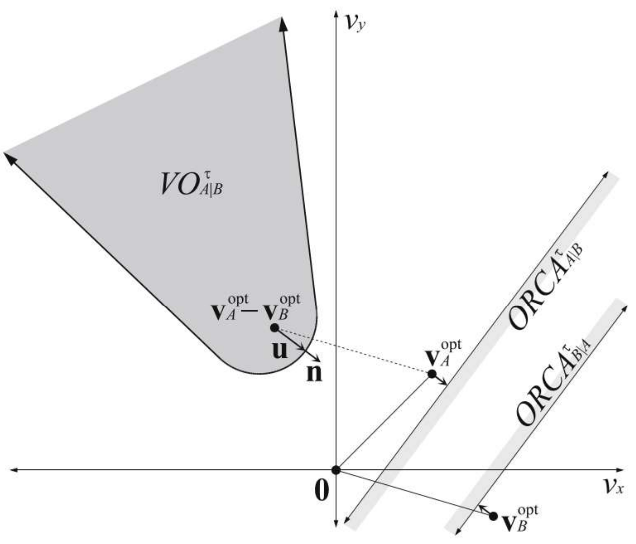
$$\lim_{\tau \to \infty} \frac{\mathbf{p}_B - \mathbf{p}_a}{\tau} = 0 \tag{2.10}$$

The complement of the Minkowksy sum $VO^{\tau}_{A|B} \oplus V_B$ is the set of collision avoiding velocities for A provided that B select its velocity from V_B which is displayed in Figure 2.13c. The definition can be found in Equation 2.11.

$$CA^{\tau}_{A|B}(V_B) = \{\mathbf{v} | \mathbf{v} \notin VO^{\tau}_{A|B} \oplus V_B\} \tag{2.11}$$

The set of velocities that is defined by the ORCA algorithm based on $CA^{\tau}_{A|B}(V_B)$ and $VO^{\tau}_{A|B}$ is computed according to Equation 2.12 and visualized in Figure 2.14. In Equation 2.12, $\mathbf{n}$ is the outward normal from the circumference of $VO^{\tau}_{A|B}$ from the position $(\mathbf{v}^{\text{opt}}_A - \mathbf{v}^{\text{opt}}_B) + \mathbf{u}$, the vector $\mathbf{u}$ represent the smallest adjustment required to the relative velocity of A and B to avoid a collision within the next τ seconds. The set of velocities $ORCA^{\tau}_{B|A}$ can be defined symmetrically. Robots A and B can deduce $ORCA^{\tau}_{A|B}$ and $ORCA^{\tau}_{B|A}$ implicitly; purely based on their own observations.

$$ORCA^{\tau}_{A|B} = \{\mathbf{v} | (\mathbf{v} - (\mathbf{v}^{\text{opt}}_A + \frac{1}{2}\mathbf{u}) \cdot \mathbf{n} \geq 0\} \tag{2.12}$$



v_y
$VO^{\tau}_{A|B}$
$\mathbf{v}^{\text{opt}}_A - \mathbf{v}^{\text{opt}}_B$
$\mathbf{u}$
$\mathbf{n}$
$\mathbf{v}^{\text{opt}}_A$
$ORCA^{\tau}_{A|B}$
$ORCA^{\tau}_{B|A}$
$\mathbf{0}$
v_x
$\mathbf{v}^{\text{opt}}_B$

Figure 2.14: Construction of velocity set $ORCA^{\tau}_{A|B}$ by addition of $\frac{1}{2}\mathbf{u}$ to $\mathbf{v}^{\text{opt}}_A$ from [94]

This principle can be extended to a scenario with n robots. Robot A senses the radius, velocity and position of the $(n-1)$ robots and computes the set of permissible velocities with respect to each intruder. The result is a set of permissible velocities for all the intersecting half-planes which is called $ORCA^{\tau}_A$. The new $\mathbf{v}_A$ is found with Linear Programming by minimizing the distance between the current velocity and the set of permissible velocities. In a very cluttered environment, it can be possible that $ORCA^{\tau}_A$ is empty and no conflict free velocity can be computed.

In [61], ORCA is implemented as a CD&R algorithm to civil aviation in which the algorithm succesfully solved potential conflicts and reduced the pilot his workload. In [62], ORCA with a of directive is implemented as a decentralized method for CD&R.

2.1.7. Conflict Prioritization

This section first explains the choice of the conflict resolution algorithm for the experimental phase. Thereafter, a description on related work in the field of conflict prioritization in conflict resolution is given.

Choice of Conflict Resolution Method

As explained in section 1.1, this research will apply conflict prioritization to an existing CD&R method based on velocity obstacles. The two considered methods are the SSD and ORCA algorithms from subsection 2.1.5 and subsection 2.1.6 respectively. Conflict prioritization in this research is represented by a binary decision. When a conflict is encountered, all aircraft from which state information is available (dependent on ADS-B constraints for manned aviation and sensory limitations for unmanned aviation) are either incorporated in the conflict resolution or completely ignored. For the SSD algorithm, this translates to activating/deactivating the VO for each aircraft in the SSD. In the ORCA algorithm this corresponds to addition/removal of constraints on the velocity space imposed by each conflict pair. Conflict prioritization is relevant for the SSD and ORCA algorithm, because these algorithms implement a joint solution which causes quick saturation of the solution space at high traffic densities yielding no feasible solution.

The SSD is selected as the method to which the conflict prioritization is first applied. In the ORCA method, each constraint introduced by a conflict pair approximately halves the solution space. In a multi-aircraft scenario with 7 or 8 aircraft, it is hypothesized that the removal of aircraft in the ORCA algorithm will result in a smaller increment in the solution space compared to the SSD algorithm. A larger solution space allows the RL algorithm to explore more options and potentially produce a more optimal solution. Moreover, comparable research with the SSD on manned and unmanned aviation is more extensive [2, 68, 16]. Hence, the SSD method is selected for this research. If the time allows it, a similar approach can be applied to the ORCA algorithm to investigate if the prioritization method generalizes beyond the SSD algorithm.

Related Work
A common approach to apply prioritization in collision avoidance is a centralized approach in which aircraft are assigned a low or high priority based on criteria such as current velocity [55, 17]. Aircraft with low priority should avoid aircraft with a high priority. Also, efforts for conflict prioritization have been made to the SSD algorithm [68]. In [68], a conflict prioritization strategy was constructed based on a fixed set of rules. With increasing priority, the rules were based on lookahead time, t_{LoS}, d_{LoS} and d_{cpa}. A sequential approach was taken which means that a conflict rule with a higher priority was selected if the current priority rule did not result in a solution. The approach showed improvement on the original SSD method. Limitations of the research were that only a limited number of rules and a limited number of combinations of those rules were evaluated.

The right-of-way (ROW) rules as part of Rules of the Air (RotA) are perhaps the most famous example of prioritization in conflict resolution [1]. The ROW can be found in Figure 2.15. It was found that following the RotA did not always reduce the complexity in airspace operations. Also, the deviation from the original flight path by complying to RotA is not minimized [58].

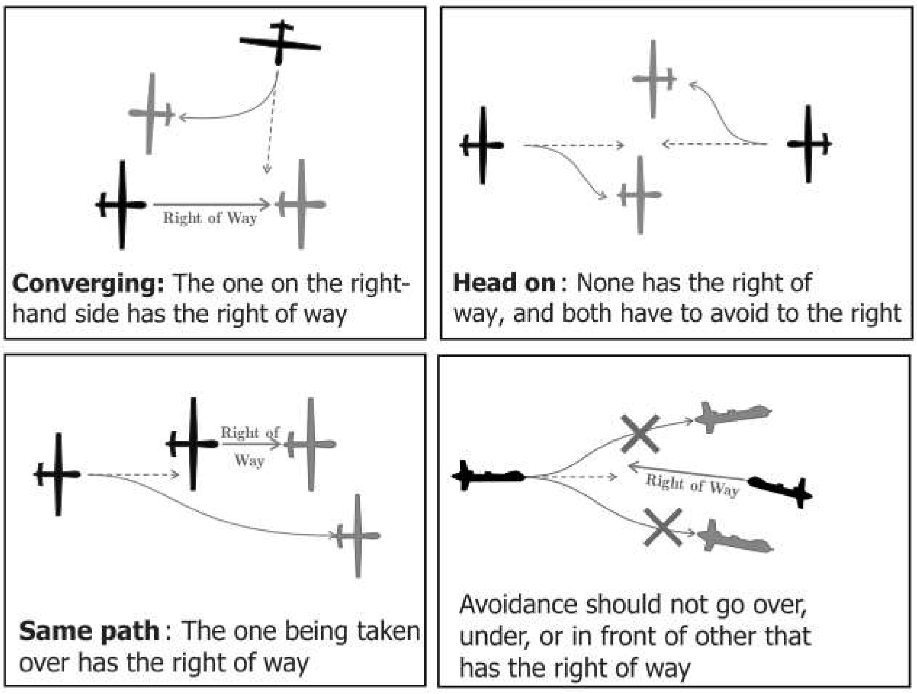

Right of Way
Converging: The one on the right-hand side has the right of way
Head on: None has the right of way, and both have to avoid to the right
Right of Way
Same path: The one being taken over has the right of way
Right of Way
Avoidance should not go over, under, or in front of other that has the right of way

Figure 2.15: The right-of-way rules from [1]

2.1.8. Dependent Variables

In this section, the definitions are provided for the variables that are measured during the experiments to assess the safety, stability and efficiency of the conflict resolution method. To accurately compare the SSD algorithm with conflict prioritization to the original SSD algorithm, the utilized evaluation metrics should be similar. The metrics are based on previous research [87, 49, 91]. An overview of the dependent variables can be found in Table 2.5. First, the parameters which measure safety are explained. The dependent variables for the efficiency and stability criteria are subsequently described.

Table 2.5: Overview of dependent variables used in the assessment of SSD algorithm as a CD&R method [2]

Variable	Type	Description
N_{conf}	Safety	Number of conflicts
N_{LoS}	Safety	Number of losses of separations
IPR	Safety	Intrusion prevention rate
LoS_{sev}	Safety	Loss of separation severity
T_{conf}	Safety	Duration of a conflict
T_{LoS}	Safety	Duration of a loss of separation
T_{inconf}	Safety	Total time in conflict per ac
T_{inLoS}	Safety	Total time in loss of separation per ac
DEP	Stability	Domino Effect Parameter
N_{mconf}	Stability	Number of aircraft involved in multi-AC conflicts
W	Efficiency	Work performed during flight
T	Efficiency	Duration of flight
D	Efficiency	Travelled distance

Safety
Safety concerns adequate separation between aircraft. Safe separation can be expressed in terms of N_{LoS} and N_{conf}. The terms LoS and intrusion indicate the same phenomenon and can be interchanged. The intrusion prevention rate (IPR) can be computed with Equation 2.13. It denotes which part of the conflicts were successfully resolved and did not result in an intrusion [87].

$$IPR = \frac{N_{cfl} - N_{LoS}}{N_{cfl}} \tag{2.13}$$

An intrusion does not directly imply a collision. Differentiation between various intrusions is achieved by two parameters: the severity of LoS LoS_{sev} and the duration T_{LoS}. For horizontal maneuvers only, LoS_{sev} can be computed with Equation 2.14 in which R represent the radius of the protected zone. The term T_{LoS} indicates the duration of a single LoS. An extension of T_{LoS} is the total time an aircraft spent in a LoS T_{inLoS}. In a similar fashion T_{conf} and T_{inconf} are defined for conflicts.

$$LoS_{sev} = \frac{R - d_{CPA}}{R} \tag{2.14}$$

Stability
At a very high traffic density, resolving conflicts can result in secondary conflicts. To measure the stability of the solution, the domino effect parameter (DEP) is used in literature [45]. The implementation of [87] for the DEP is chosen and is defined by Equation 2.15. In Equation 2.15, S_1 represents the subset of conflicts if CR is deactivated during simulation and S_2 the subset of conflicts if CR is activated. A high DEP thus implies that the CR method often incites secondary conflicts. The number of aircraft which experience multi-aircraft conflicts N_{mconf} is a second measure for stability.

$$DEP = \frac{S_2}{S_1} - 1 \tag{2.15}$$

Efficiency
A third objective for which conflict resolution methods can be optimized is efficiency. Efficiency is measured with the work done during a flight W, the flight time T and the length of the traversed flight path D. The work W is defined with Equation 2.16 in which $\mathbf{T}$ and $\mathbf{s}$ denote a thrust and displacement vector respectively.

$$W = \int_{\text{path}} \mathbf{T} \cdot \mathbf{ds} \tag{2.16}$$

2.2. Reinforcement Learning Basics

The basic concepts of reinforcement learning which are required to understand the principles on which the advanced methods are based, are explained in this section. First an introduction to the topic of reinforcement learning is provided in subsection 2.2.1. Thereafter, a description of the research methodology is provided in subsection 2.2.2. A taxonomy of RL methods is given in subsection 2.2.3. Thereafter, the basic mathematical framework for RL problems is explained in subsection 2.2.4. Subsequently, the concepts of reward, policy and value function are elaborated upon in subsection 2.2.5. Section 2.2.6 describes how the optimal policy can be found. In subsection 2.2.7, the difference between a model-free and model-based approach is explained. Finally, subsection 2.2.8 and subsection 2.2.9 describe value- and policy based methods.

2.2.1. Introduction

A gazelle calf is barely able to stand on its feet just after it is born. However, within hours it is able to run around. In the first couple of hours of its life, the gazelle continuously receives feedback from its environment on whether the movements of its limbs successfully contribute to the process of being able to stand up and run around. The gazelle is able to learn which movements are successful and incrementally becomes better at moving around. This is an example of how an agent (the gazelle) with no previous information, is able to determine optimal behaviour by interacting with the environment.

The term Artificial Intelligence (AI) is a general term which already originates from 1956 [24] and encompasses a great variety in algorithms and methods. Artificial intelligence is defined as "a system's ability to interpret external data correctly, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation" [43]. The branch of AI that focuses on problems similar in nature to the gazelle calf that learns to walk by interacting with the environment without an explicit teacher, is called Reinforcement Learning (RL) [88].

The agent and environment are the two main elements that each RL problem has. The environment is the world as the agent sees it and the state is an observation of how an agent perceives the environment at a specific time. An environment can be altered by the action of an agent or change on its own. The agent observes a reward from the environment which is essentially a feedback mechanism that provides a numerical value to how desirable the selected action of the agent was based on the future environment state.

The crux in reinforcement learning lies in how the agent selects an action based on an observation of the environment. To define the scope of the examined methods in this chapter, a taxonomy is provided in subsection 2.2.3.

2.2.2. Methodology

In this section, a motivation is provided on why the included sections are relevant in the research to decide on the type of reinforcement learning model. The basic concepts of RL within the relevant scope are explained in section 2.2. The relevant scope will be defined in subsection 2.2.3. Thereafter, function approximators and challenges on MARL are discussed in subsection 2.3.1 and subsection 2.3.3 respectively. The sections are relevant, because almost all state-of-the-art methods utilize function approximators and the proposed objective of applying conflict prioritization to multiple aircraft is a multi-agent problem. Relevant literature in subsection 2.3.5 is described with a focus on RL model type and the methodologies to deal with the challenges imposed by multi-agent RL. The advanced algorithms are related to previous section, because they implement function approximators and their inner mechanics can be explained with aid of the basic concepts. The relevant advanced algorithms are further explained in detail in subsection 2.3.6 before a decision about the final method is made in subsection 2.3.7.

2.2.3. Taxonomoy

Due to extensive research, a great variety of reinforcement learning algorithms has been established. The properties of the problem determine which algorithms are most suitable. An abbreviated taxonomy is provided in Figure 2.16 which assists to narrow the scope of this research to algorithms that could actually be implemented to solve the problem imposed in chapter 1. The first division is between algorithms that use finite Markov Decision Processes (MDP) and bandits. This research focuses on MDPs.

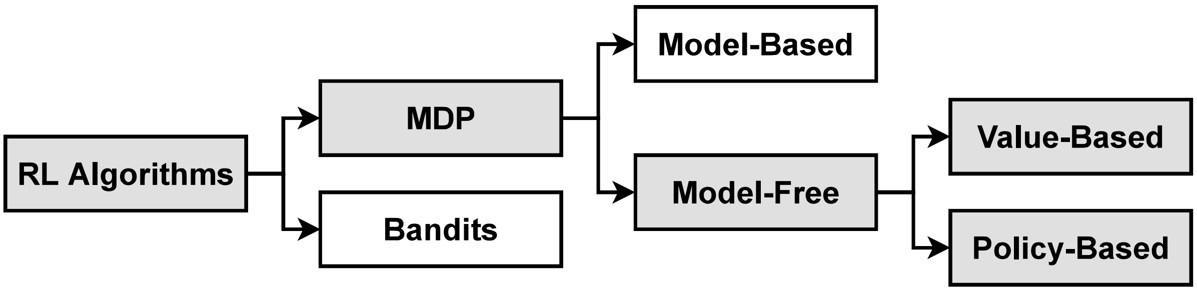


Figure 2.16: RL taxonomy adapated from [101]

2.2.4. Markov Decision Processes

A MDP is a mathematical framework in sequential decision making [88]. It does not only take the immediate reward based on an action into account, but also subsequent states and thus rewards. A Markov Decision Process in the context of RL can be explained with Figure 2.17. At time $t = 0$, agent receives state $s_0 \in S$ and takes action $a_0 \in A$ with S and A being the sets of all valid states and actions respectively. Applying action a_0 to the environment results in $s_1 \in S$ and $r_1 \in R$ with R as the set of all valid rewards. The continuous interaction between agents and environment for n timesteps into the future results in a trajectory τ represented by Equation 2.17.

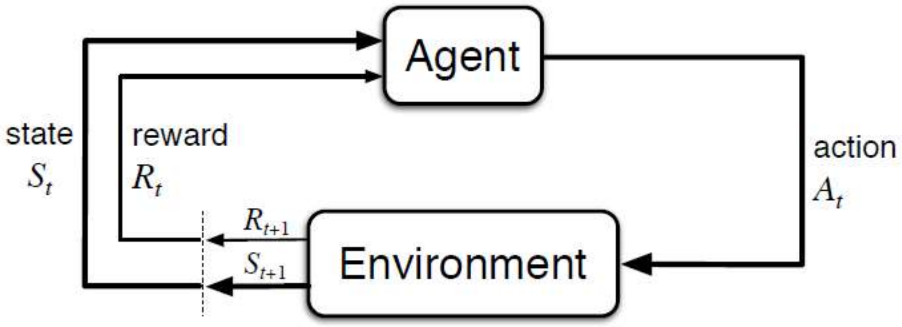


Figure 2.17: Interaction loop between agent and environment

$$\tau = s_0, a_0, s_1, r_1, a_1, s_2, r_2, a_2, ...s_n, r_n, a_n. \tag{2.17}$$

If the series of Equation 2.17 is a Markov decision process, the system dynamics are completely determined by Equation 2.18. That means that the probability, for all valid s_t and r_t, is solely dependent on the last state and action, namely s_{t-1} and a_{t-1}. If the state is able to capture all relevant information about past interaction in the previous state, the Markov Property can be assigned to the state. The Markov property is assumed to be true for the rest of this research.

$$p\left(s', r \mid s, a\right) = \Pr\left\{s_t = s', r_t = r \mid s_{t-1} = s, a_{t-1} = a\right\} \tag{2.18}$$

2.2.5. Reward, Policy and Value Function

For optimal behaviour an agent should not only focus on immediate rewards, but also on future rewards. There are RL problems that can easily be divided in sequences in which the agent selects a new action until the terminal state is reached. Once the terminal state is reached, a new sequence is started. Those sequences are called episodes. An example of an episode is playing an entire game of Pac-man. Maximizing the reward for such problems could simply mean to maximize the total sum of all rewards received within one episode. However, there are also continuous RL problems in which time runs to infinity which obviously would make this simple approach not very attractive since the sum of rewards might not converge. Therefore, the concept of discounting is introduced which results in a new function that the agent tries to optimize. This new function is called the total discounted reward G_t or return and is represented by Equation 3.8.

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{2.19}$$

In Equation 3.8, γ (with $0 \leq \gamma \leq 1$), denotes the discount factor which reduces the effect of future rewards and increases the effect of more recent reward.

The value function is incorporated in almost every reinforcement learning algorithm and indicates the long term value of being in a state. It is an indication for the agent on how desirable the current state is. A policy assigns a probability to all possible actions that an agent can take and fully defines the behaviour of an agent. An agent with policy π has probability $\pi(a \mid s)$ of selecting action a given state s. Equation 2.20 displays this relationship in a more formal manner. It should be noted that a policy is time-invariant [83].

$$\pi(a \mid s) = \mathbb{P}\left[a_t = a \mid s_t = s\right] \tag{2.20}$$

Given a policy π, the value function of a state $v_\pi(s)$ can be defined according to Equation 2.21 for MDPs. The value function is the expected return when starting in state s and subsequently following policy π. The function $v_\pi(s)$ is called the state-value function for policy π.

$$v_\pi(s) = \mathbb{E}_\pi\left[G_t \mid s_t = s\right] \tag{2.21}$$

In a similar fashion, the action-value function for policy π can be determined. The action-value function $q_\pi(s,a)$ represents the value of selecting value a in state s while following policy π and can be found in Equation 2.22.

$$q_\pi(s,a) = \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right] \tag{2.22}$$

A very fundamental equation in RL is the Bellman equation. The Bellman equation describes the value of a current state by taking the expected value of the immediate reward and the discounted sum of the value function for the subsequent state. The basic mathematical idea behind an expected value computation is to multiply every possible value that a random variable can take on with the probability of occurrence and sum all values. For a discrete random variable X, this computational process can be found in Equation 2.23 with $f(X)$ being the probability density function.

$$\mathbb{E}[X] = \sum X_i \cdot f(X_i) \tag{2.23}$$

A similar approach can be applied to the first line of Equation 2.24. The policy determines the probability $\pi(a \mid s)$ for action a given state s. Action a has probability $p\left(s', r \mid s, a\right)$ to transform the state of the agent from s to s' given a. State s' leads to reward r and discounted value $\left(v_\pi\left(s'\right)\right)$. So the probability of reaching state s' is equal to $\pi(a \mid s) \cdot p\left(s', r \mid s, a\right)$ and the value of s' is $\left(r + v_\pi\left(s'\right)\right)$. Summing over all possible states s' based on action a and all actions a results in the recursive algorithm of Equation 2.24.

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi\left[r_{t+1} + \gamma G_{t+1} \mid s_t = s\right] \\ &= \sum_a \pi(a \mid s) \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma v_\pi\left(s'\right)\right], \quad \text{for all } s \in S \end{aligned} \tag{2.24}$$

A similar derivation can be made for the action-value function $q_\pi(s,a)$ resulting in Equation 2.25.

$$q_\pi(s,a) = \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma \sum_{a'} \pi\left(a' \mid s'\right) q_\pi\left(s', a'\right)\right] \tag{2.25}$$

2.2.6. Optimal Policies

An optimal policy is denoted as π_* and this means that there is no policy with a higher state-value function for all states. There can, however, be multiple optimal policies (e.g. multiple trajectories with similar rewards could lead to multiple policies which are equally optimal). The optimal state-value function $v_*(s)$ is defined by Equation 2.26.

$$v_*(s) = \max_\pi v_\pi(s) \tag{2.26}$$

The optimal action-value function $q_*(s,a)$ is displayed by Equation 2.27 and denotes the maximum action-value function over all possible policies. The optimal policy is followed after starting in state s and taking action a [83]. Subsequently, the optimal policy is followed.

$$q_*(s,a) = \max_{\pi} q_\pi(s,a) \tag{2.27}$$

The optimal state-value function $v_*(s)$ and the optimal action-value function $q_*(s,a)$ are related to each other by the Bellman optimality equation (Equation 2.28). Intuitively, this makes sense, because it essentially says that when in state s, select the action that maximizes the optimal action-value function $q_*(s,a)$. It is known that $q_*(s,a)$ follows the optimal policy after selecting action a and hence if state a is also optimal with respect to $q_*(s,a)$, the optimal policy is followed resulting in an optimal value function. [88]

Comparing Equation 2.28 with Equation 2.24, the discrepancy can be found in the first part of the equation. Instead of assigning a probability a to each possible action from state s based on the policy $\pi(a \mid s)$, a 100% probability is assigned to the action which maximizes the sum of immediate reward and discounted value function based on future states.

$$\begin{aligned} v_*(s) &= \max_{a \in A} q_{\pi_*}(s,a) \\ &= \max_{a} \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma v_*\left(s'\right)\right] \end{aligned} \tag{2.28}$$

The Bellman equation for the optimal action-value function $q_*(s,a)$ can be found in Equation 2.28. Again, compared to Equation 2.25, the summation over all possible actions is exchanged for the action which maximizes $q_*(s,a)$.

$$q_*(s,a) = \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma \max_{a'} q_*\left(s', a'\right)\right] \tag{2.29}$$

The goal is to compute the optimal policy $\pi_*(a \mid s)$. The policy allows the agent to maneuver through the environment. Hence, with an optimal policy, the agent knows the most desirable way to maneuver. The optimal policy can be derived from $v_*(s)$ or $q_*(s,a)$.

To derive the optimal policy from $v_*(s)$, the action should be selected that is optimal in the next step. Due to the definition of the optimal state value function, performing this method for each singular step actually results in the optimal policy. The policy is said to be greedy with respect to the value function. An example can be found in Figure 2.18, in which the values of possible future states are visualized. The optimal policy would select action a_2 because it results in the state with the highest value of all actions a_1, a_2, a_3 and a_4. With the optimal $q_*(s)$, this process is even more effortless and the action that maximizes $q_*(s)$ is selected. More formally, this is defined by Equation 2.30 [83].

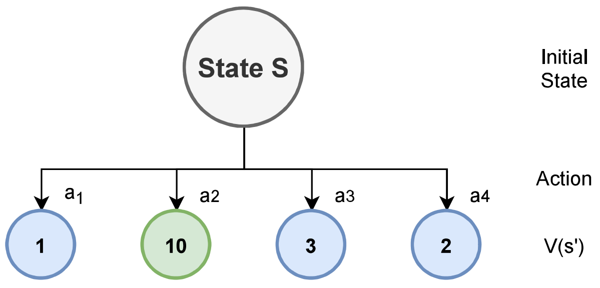


Figure 2.18: Optimal policy would select action a_2 based on state S

$$\pi_*(a \mid s) = \begin{cases} 1 & \text{if } a = \underset{a \in A}{\operatorname{argmax}}\left(q_*(s,a)\right) \\ 0 & \text{otherwise} \end{cases} \tag{2.30}$$

2.2.7. Model-Based vs Model-Free

From Figure 2.16, it can be seen that the category of MDP consists of two subcategories namely Model-Based and Model-Free. Model-Based and Model-Free refer to the availability of the exact model dynamics as defined by Equation 2.18. In this research, the focus will lie on the Model-Free methods since the exact model dynamics are not known in the case of the multi-agent collision resolution scenario to which the RL algorithm will be applied. A famous example of an agent that does incorporate a model-based approach is the AlphaGo from DeepMind [85] mentioned in chapter 1.

The Model-Free algorithms can learn from experience. Experience are states, actions and rewards that are accumulated by interacting with the environment. From the taxonomy in Figure 2.16, it can be seen that the Model-Free methods are built up from the subcategories: value-based and policy-based.

2.2.8. Value-Based Methods

The difference between on-policy Methods and off-policy methods lies in how the policy is used to find the optimal policy. In on-policy methods the policy which determines which action the agent has to take, the *behavioural policy*, is the same policy that is optimized by the RL algorithm, the *target policy*. In off-policy method the behavioural policy is different than the target policy. The experience collected by the behavioural policy is utilized by the agent to update the target policy.

When learning from experience with a Model-Free method, it is important that the algorithm allows for sufficient exploration of actions and states. This is important to prevent the algorithm from immediately selecting the actions that initially seem most optimal, because it is possible that other actions, which are actually more optimal, are never selected. On-policy and off-policy methods ensure exploration in a different manner. Off-policy methods ensure proper exploration by implementing a behavioural and target policy.

In on-policy methods, use is often made of a ϵ-greedy policy which means that a nonzero probability is assigned to all actions a in each state s. In a ϵ-greedy policy, which encourages exploration, all actions m are selected with non-zero probability ϵ/m. An extra probability of $1-\epsilon$ is assigned to the greedy action. but with probability ϵ one of the non-optimal actions is selected. The policy is represented by Equation 2.31. This policy allows for exploration by implementing a non-optimal policy and ϵ is a hyperparameter which can be manually tuned [83].

$$\pi(a \mid s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a^* = \underset{a\in A}{\text{argmax}} Q(s,a) \\ \epsilon/m & \text{otherwise} \end{cases} \tag{2.31}$$

On-Policy Method: SARSA

A well-known on-policy method is called SARSA. The SARSA-algorithm updates the action-value function following the logic of Equation 2.32 in which α is the learning rate. It can be seen that SARSA utilizes two state-action pairs and one reward resulting in the set: $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$. It immediately becomes evident where the name SARSA originates from. The value function $Q(S_t, A_t)$ is updated with a known reward R_{t+1} and with an estimate of the value function $Q(S_{t+1}, A_{t+1}$. Updating an estimate with an estimate is called *bootstrapping*. The actions A_t and A_{t+1} on the right-hand side of Equation 2.32 are from the current policy and are utilized to update the current policy (left-hand side).

$$Q(s_t, a_t) \leftarrow Q(a_t, a_t) + \alpha \left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right] \tag{2.32}$$

Off-Policy Method: Q-learning

An algorithm which was already developed in 1989 and is still widely applied is called *Q-learning* [97]. Q-learning is an off-policy method and the action-value function is updated according to Equation 2.33. The action-value function $Q(S_t, A_t)$ is updated in a way such that it approximates the optimal action-value function. This is accomplished by the max operator which ensures that the action a in state S_{t+1} is chosen that maximizes the current approximation of the action-value function. The action a is thus chosen independently of the current behavioural policy.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \tag{2.33}$$

2.2.9. Policy Gradient Methods

Besides algorithms that try to learn the optimal action-value function and base the optimal policy on it, a different branch of algorithms exists that focuses on optimizing the policy without requiring a value function. The policy is a *parameterized policy* of which the derivatives with respect to its parameters θ should exist. The notation $\pi(a \mid s,\theta)$ or π_θ means that the policy is dependent on parameter θ. The goal of policy based methods is to optimize the parameters settings such that an objective function $J(\theta)$ is maximized. For a stochastic policy, the objective function $J(\theta)$ is equal to the expected total reward $\sum_{t=0}^{\infty} \gamma^t R_t$ [76].

The updates of the parameter settings are performed according to Equation 2.35 [88] in which α is parameter that determines the step size. The gradient of the objective function with respect to the policy parameters is added to the current parameter settings θ_t. This makes sense, because say that the derivative of the objective function with respect to parameter θ_1 is 1 (Equation 2.34), an increase in the parameter θ_1 will cause an increase in the objective function which means a more positive value should be assigned to θ_1 and that is exactly what happens. If the derivative is negative, the exact opposite holds: a decrease in the parameter value will cause an increase in the objective function and a smaller value for θ_1 is thus desired.

$$\frac{\delta J(\theta)}{\delta \theta_{t,1}} = 1 \tag{2.34}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \widehat{\nabla J(\boldsymbol{\theta}_t)} \tag{2.35}$$

The term $\widehat{\nabla J(\boldsymbol{\theta}_t)}$ is a stochastic estimate of the policy gradient g. A stochastic estimate means that the gradient g is not computed on the entire data set, but on a sample or batch of samples. The policy gradient g can be computed in a variety of ways and have the general form of Equation 2.36 [76]. The different representations have the same expected value, but differ in the amount of variance.

$$g = \hat{\mathbb{E}}\left[\sum_{t=0}^{\infty} \Psi_t \nabla_\theta \log \pi_\theta (a_t \mid s_t)\right] \tag{2.36}$$

Where ψ_t can be any of the following choices in which the discount factor γ is assumed to be 1 [76]:

1. $\sum_{t=0}^{\infty} r_t$: cumulative reward of trajectory.
2. $\sum_{t'=t}^{\infty} r_{t'}$: immediate reward by following action a_t.
3. $\sum_{t'=t}^{\infty} r_{t'} - b(s_t)$: Similar to 2, but with a baseline.
4. $Q^\pi(s_t, a_t)$: state-action value function.
5. $A^\pi(s_t, a_t)$: as an advantage function.
6. $r_t + V^\pi(s_{t+1}) - V^\pi(s_t)$: Temporal Difference residual.

To illustrate the general problem of the bias-variance tradeoff in machine learning and explain why it is possible that all different options for ψ_t have the same expected value, Figure 2.19 is utilized. The left image showcases an approach with high bias, but no variance. In the middle image, it can be seen that there exists no bias, but a high variance. Finally, the image on the right shows a trade-off between variance and bias which is often desired. All three situations showcase different amounts of bias and variance, but have a similar expected value. In the context of policy gradient algorithms high variance among samples impedes convergence. One sample could adapt the weights in one direction and if the next sample is very different and adapts the weights in the opposite direction, one can imagine that convergence is difficult. Introducing for example a baseline function (Option 3) reduces the variance, but increases the bias.

The advantage function is defined according to Equation 2.37 [76]. The advantage function measures how good a certain action is compared to following the current policy.

$$A^\pi(s_t, a_t) := Q^\pi(s_t, a_t) - V^\pi(s_t) \tag{2.37}$$

The advantage function is often not available and has to be estimated. An estimator that is widely implemented along policy gradients algorithms is called the generalized advantage estimator (GAE) [76]. The bias-variance trade-off is influenced by two parameters in GAE namely λ and γ when an approximate value function is utilized. A general value for $\gamma = 0.99$ and for λ in the somewhere in the range $[0.9, 0.99]$.

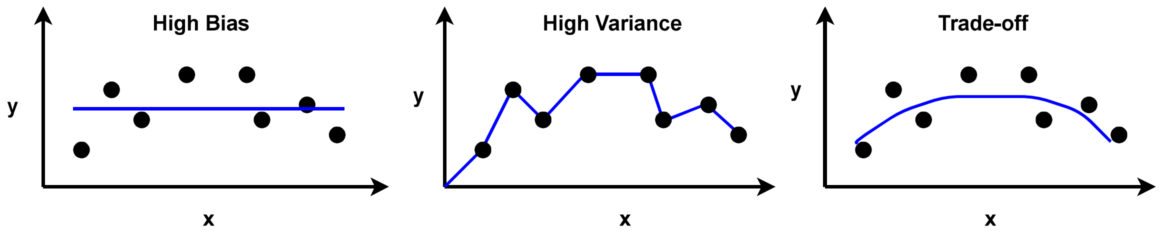


Figure 2.19: The bias-variance trade-off

2.3. Advanced Reinforcement Learning

State-of-the art reinforcement learning methods are discussed in this section. First, function-approximatos are explained in subsection 2.3.1. Thereafter an introduction to multi-agent reinforcement learning and the accompanying challenges is provided in subsection 2.3.2 and subsection 2.3.3. The various types of control schemes in MARL are described in subsection 2.3.4. Within the defined scope of RL algorithms, solutions to MARL problems from relevant research are given in subsection 2.3.5. The RL models from literature are explained in subsection 2.3.6. Finally, in subsection 2.3.7 a discussion is provided on the most optimal model for this research for which details on the implementation are given in subsection 2.3.8.

2.3.1. Function Approximation

Up untill this point we dealt with tabular approach of Q(s,a) and V(s). Tabular methods become impractical for a large state and action space. Instead, function approximations methods are utilized. The approximation function can for example be a linear function approximation in which the function value is equal to a feature vector multiplied with weights. Even though the linear approximation function is shown to theoretically always converge, its inability to capture complex non-linear relationships meant that the search to other function approximators continued [88]. A notable breakthrough was a paper in which a convolutional Neural Network (CNN) was implemented to estimate the action-value function with raw video pixels as state input [36]. The algorithm is called Deep Q-Learning (DQN) and showed performance which exceeded the human expert when playing Atari 2600 games [3].

To stabilize the Q-learning process with a non-linear function approximator ($Q(s,a;\theta) \approx Q(s,a)$), use was made of a replay memory buffer and a fixed target network. The replay buffer entails that the training data is randomly sampled from a buffer of experiences to tackle the problem of correlated data and non-stationary distributions [52]. It is also increases the data efficiency since one sample is utilized multiple times. Note that in practice, the experience buffer has a limited size, so if the buffer is full, the oldest experiences are removed. The replay memory buffer consists of observations in the format (s_t, a_t, r_t, s_{t+1}). The loss function of the DQN algorithm can be found in Equation 2.38 in which y_i is the target value for the i^{th} iteration. The parameters θ_i are optimized such that the loss function is minimized based on a batch of samples from the experience replay buffer. At iteration $i+1$, $Q(s,a;\theta)$ is optimized to approximate the target network. If the target network shares the same weights then the target network is also updated causing a chase to a moving target. Therefore, the weights of the target network are fixed and updated every C iterations [36].

$$L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}\left[\left(y_i - Q(s,a;\theta_i)\right)^2\right] \tag{2.38}$$

Artificial Neural Network

The CNN implemented in the DQN paper [36] is an Artificial Neural Network (ANN) with multiple layers making it a Deep Neural Network (DNN) [50]. The input of a CNN is usually an image. The specific architecture and working principle of a CNN will not be elaborated upon since the input state in this research will not be an image. A deep network refers to the existence of multiple layers in its model that learn representation of the data in an abstract manner. The most common implementation of deep learning algorithms is in a supervised learning fashion. This also holds for implementing the deep models in a reinforcement learning setting. In supervised learning the desired output of the model is known and the model weights are optimized to reduce the error between the model output and the desired model output. Equation 2.38 is a good example of the loss function for a supervised learning problem. The desired output is the target y_i and the weights θ are adapted such that $Q(s,a;\theta_i)$ better approximates y_i.

The general structure of a multi-layer NN consists of an input layer, n hidden layers and an output layer with weights connecting the nodes between layers. This can be seen in the left image of Figure 2.20. The input values at the the first hidden layer H1 are computed by a weighted sum of the output values of the connected nodes from the previous layer. In the hidden layer H1 the input is transformed by a non-linear activation function (bias is left out to simplify the situation). The rectified linear unit (ReLU) is popular and equals $f(z) = \max(z, 0)$, but alternatives such as the $\tanh(z) = \frac{1}{1+\exp(-z)}$ are also often used. This computational process is repeated for each layer until an output is computed at the output layer. Transforming the input of the network to the output is referred to as the forward pass.

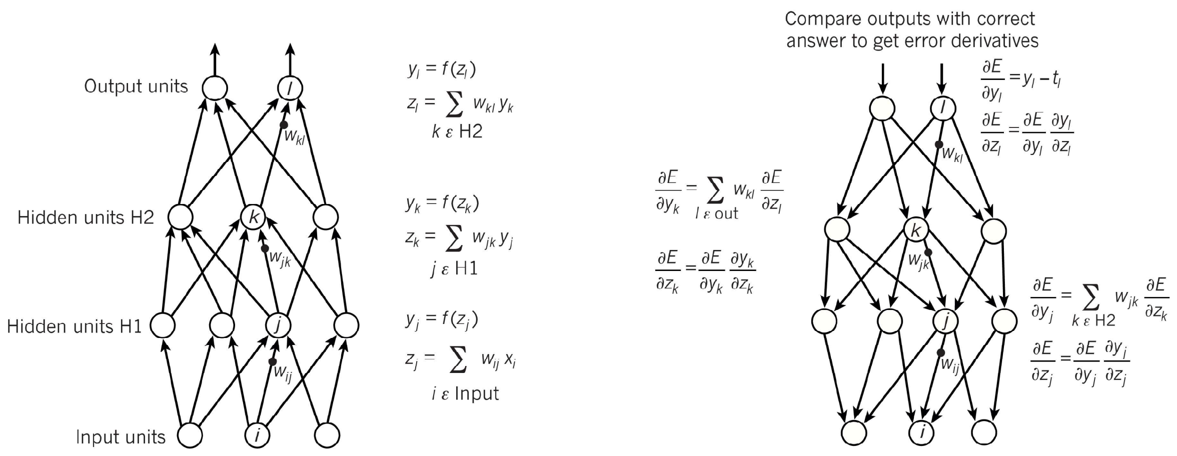

Output units
Hidden units H2
Hidden units H1
Input units
l
k
j
i
w_{kl}
w_{jk}
w_{ij}
$y_l = f(z_l)$
$z_l = \sum_{k \in H2} w_{kl} y_k$
$y_k = f(z_k)$
$z_k = \sum_{j \in H1} w_{jk} y_j$
$y_j = f(z_j)$
$z_j = \sum_{i \in Input} w_{ij} x_i$
Compare outputs with correct answer to get error derivatives
$\frac{\partial E}{\partial y_l} = y_l - t_l$
$\frac{\partial E}{\partial z_l} = \frac{\partial E}{\partial y_l}\frac{\partial y_l}{\partial z_l}$
$\frac{\partial E}{\partial y_k} = \sum_{l \in out} w_{kl} \frac{\partial E}{\partial z_l}$
$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial z_k}$
$\frac{\partial E}{\partial y_j} = \sum_{k \in H2} w_{jk} \frac{\partial E}{\partial z_k}$
$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_j}$

Figure 2.20: Schematic representation of deep neural network in which the forward and backward pass are visualized

With the general structure and working principle of a deep network explained, an elaboration on how the weights will be updated will be provided with aid of the right figure in Figure 2.20. The cost function C in the example is Equation 2.39 [50], in which y_l is the output layer value and t_l is the target value. The goal is to compute a function in which the derivative of the cost function with respect to all network weights is known. The computational process for the derivative of the error with respect to w_{jk}, which is the weight connecting neuron j with neuron k, is described. The gradient of the cost function with respect to the network parameters can be obtained by applying the chain rule for derivatives. The derivation starts with Equation 2.39 and z and y refer to the input and output of a layer respectively.

$$E = \frac{1}{2}(y_l - t_l)^2 \tag{2.39}$$

The partial derivative of Equation 2.39 with respect to y_l is simply $(y_l - t_l)\frac{\delta E}{\delta z_l}$. The derivative $\frac{\delta E}{\delta z_l}$ can be found by multiplying $\frac{\delta E}{\delta y_l}$ with the derivative of the activation function $\frac{\delta y_l}{\delta z_l}$. The derivative $\frac{\delta E}{\delta z_k}$ can be found with Equation 2.40. Since node k is connected to both output nodes, a change in y_k influences the output in both nodes and thus also the error in both nodes. Therefore, the partial derivative $\frac{\partial E}{\partial y_k}$ is a summation of errors terms each belonging to a path from which y_k can influence E.

$$\frac{\partial E}{\partial y_k} = \sum_{l \in \text{out}} w_{kl} \frac{\partial E}{\partial z_l} \tag{2.40}$$

In a similar fashion, the derivatives for z and y with respect to the cost function can be constructed for the other network layers up until the input layer. The derivative $\frac{\delta E}{\delta w_{jk}}$ can be be constructed by multiplying y_j with

$\frac{\delta E}{\delta z_k}$ which is the desired end result. To start at the output and compute the derivatives of the cost function with respect to the weights from there, is called backpropagation. With aid of backpropagation, weights are updated according to Equation 2.41 in which θ_i is the updatable weight and α the learning rate. The algorithm is called stochastic gradient descent (SGD) in the context of RL, because the updates of the weights can never be made on a "complete" data set since the agent continues to explore the environment. It would be called gradient descent (GD) if the update of the weights is based on all data samples from the train set.

$$\theta_{t+1} = \theta_t - \alpha\left(\frac{\delta E}{\delta \theta_i}\right) \tag{2.41}$$

Gradient Descent Optimization Algorithms

Stochastic gradient descent suffers from multiple issues [75] such as:

- Tuning of the hyperparameter α can be tedious.
- Using a fixed learning rate schedule [74] on which the magnitude of the learning rate is decreased prevents adaption of the algorithm to specific characteristics of the data set.
- The same learning rate is applied to all parameters. Weights that belong to rare features might want to be updated with a larger step than weights belonging to features that occur in many input data samples.
- Getting trapped in local minima in a non-smooth optimisation landscape.

A variety of gradient descent optimization algorithm exist. Which optimizer is most optimal is correlated to type of optimisation problem. In general, the Adaptive Momentum Estimation (Adam) Optimizer is most optimal and the hyperparameters typically do not require extension tuning [75]. The Adam optimizer can be seen as a successor of the Adagrad [15] and RMSprop [90] optimization algorithms. The Adagrad algorithm can handle sparse features well and RMSprop is more suitable for on-line parameter updates in non-stationary environments [44]. Adam is an optimisation method which assigns an adaptive learning rate to each individual model parameter while also implementing a momentum based gradient. The parameters are updated according Equation 2.42.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2.42}$$

in which ϵ is a term which prevents a division by zero, η is the learning rate, $\hat{m}_t$ is the biased corrected first moment estimate and $\hat{v}_t$ bias corrected second order estimate. Default values for ϵ and η are 10^{-8} and 0.001 respectively [44].

The term $\hat{m}_t$ is computed by taking an exponentially weighted moving average of past gradients in a fixed time window w. The idea of replacing the gradient with a weighted moving average of past gradients originates from the momentum gradient descent optimisation algorithm. It improves the convergence of the SGD in the desired direction and reduces a fluctuating behaviour around the desired gradient direction.

The learning rate η is divided by $\sqrt{\hat{v}_t} + \epsilon$ causing an adaptive learning rate for each parameter. $\hat{v}_t$ is found by taking a bias corrected exponentially weighted moving average of past squared gradients in a fixed time window w. The term $\hat{v}_t$ represents an estimation of the variance in the gradients.

Recurrent Neural Networks

A different type of ANN which is suitable for sequential input data such as speech or text is a recurrent neural network (RNN). RNNs can also be implemented as a policy in reinforcement learning for a wide variety of problems [46]. A RNN is a network with loops which allows information to be transferred in a hidden state such that information about the history of input data can be used at the current input. This process is visualized in Figure 2.21 in which x_t and h_t represent the input and output of the RNN module at time t. If the looped network is unfolded, it can be seen as a chain of identical neural networks which transfer information to each other through the hidden state.

Regular RNNs experience the problem of vanishing or exploding gradients when the back propagation algorithm is implemented for a large number of time steps [50] causing difficulty for learning long term dependencies in the input data. A special kind of RNN solves this problem namely a Long Short Term Memory (LSTM) Network. The original idea of the LSTM dates back from 1997 when it was created by Hochreiter Schmidhuber [30].

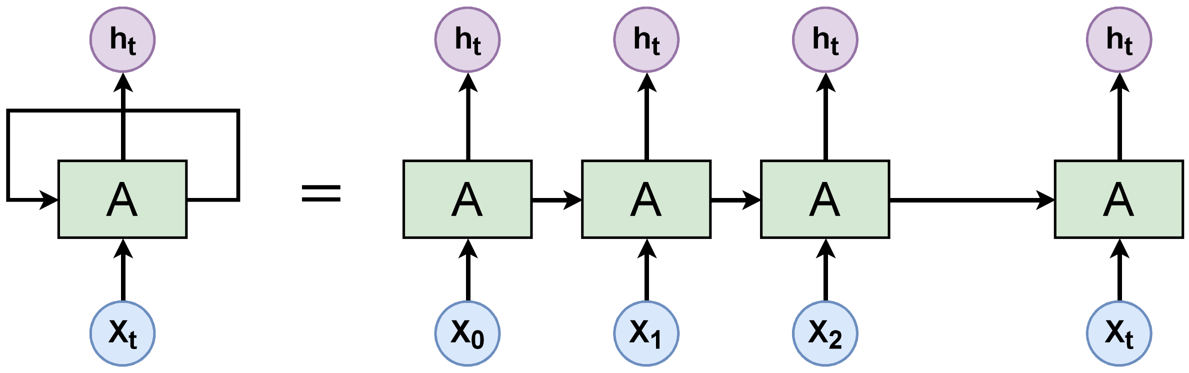
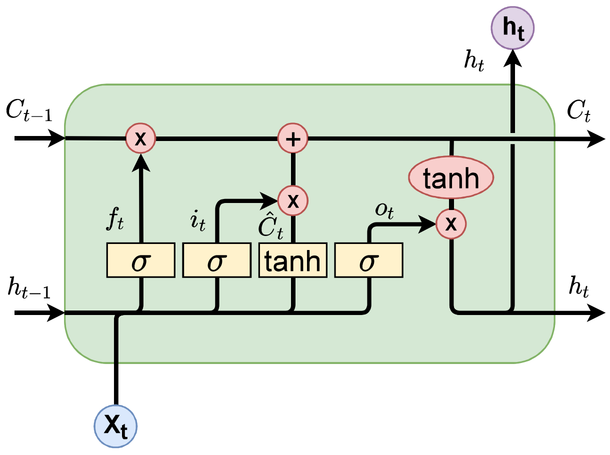


Figure 2.21: Unrolled LSTM network adapted from [63]

The structure of an LSTM cell is displayed in Figure 2.22 [63]. The idea of an LSTM is to to control the cell state C_t which can be found at the top of the cell. The cell state is controlled by three gates. The gates are called the forget gate, input gate and output gate. All gate layers take the input at time t x_t (input sample at time t) and the h_{t-1} (output of network at time $t-1$) as input. The forget gate layer outputs f_t which takes on a value between 0 and 1 and describes for each element in the cell state how much of the information needs to be remembered. The input layer gate outputs i_t which determines which parts of the new information which can be added to the cell state $\tilde{C}_t$ (partly based on the new input sample x_t) are added to the cell state. The output layer gate output o_t is pointwise multiplied with the cell state and determines which parts of the cell state are provided as an output. The mathematical definitions of f_t, i_t, $\tilde{C}_t$, C_t, o_t and h_t can be found in Equation 2.44-Equation 2.48 in which W and b denote the weights and bias for each layer.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{2.43}$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{2.44}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{2.45}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.46}$$

$$o_t = \sigma\left(W_o\,[h_{t-1}, x_t] + b_o\right) \tag{2.47}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{2.48}$$

Figure 2.22: Cell structure of LSTM network adapted from [63]

Hyperparameter Tuning

Hyperparameters are the parameters which are set before the training is started. Examples of hyperparameters can be the number of nodes in the hidden layer or the discount factor γ. Hyperparameter tuning is a tedious process, but important to obtain a well converging algorithm. Four common methods to select the optimal set of hyperparameters are:

- **Manual Tuning**: Hyperparameters are selected by hand. This requires intuition from the user and becomes increasingly more difficult with increasing number of parameters and the ranges of values which those parameters can attain [98].
- **Grid Search**: Based on exhaustive search. The hyperparameter ranges are discretized and a model is trained for each possible combination of hyperparameters. The set of hyperparameters with the highest performing model is selected. A grid search is accompanied with a high computational load, because it suffers from the curse of dimensionality [4].
- **Random Search**: This is a variation on grid search. Random search reduces the number of hyperparameters by focussing on the hyperparameters that have the greatest influence on performance [4].
- **Bayesian Optimization**: The optimization function in the hyperparameter tuning process is a black-box function. Bayesian optimization is suitable for an optimization problem with these characteristics. The Bayesian formula is utilized to obtain information on where the objective function is maximized [98].

2.3.2. Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) is a branch within RL in which a system of autonomous agents is simultaneously interacting in a common environment [54]. The theory explained up until this point has mostly been with regard to reinforcement learning in a single-agent setting. This means that the actions of the policy only influence one agent in the environment. The goal of this thesis is to improve CD&R in a high traffic density scenario in which multiple aircraft implement conflict prioritization. This is a multi-agent setting. Therefore, this section aims to explain how the concepts of the single-agent settings are related to the multi-agent setting and which new problems and concepts are introduced due to this increase in number of interacting agents.

Mathematical Framework

An extension of the MDP mathematical framework defined for the single-agent RL case to the multi-agent case is called a Markov Game [53]. A Markov Game in its most generic form can be described with a tuple $\langle N, S, A, R, P, \gamma \rangle$ [54] in which S is a set of all possible states which describe the agents in the environment, $A = \{A_1, A_2, ...A_N\}$ is a set of actions for all agents N, R is the reward function, P denotes the state transition probability based on current state plus action from each agent: $S \times A_1 \times A_2 \times A_N \rightarrow$ P(s' | s,a) and γ denotes the discount factor. Note that each agent i has their own reward function R_i: $S \times A_1 \times A_2 \times A_N \rightarrow \Re$ and strives to maximize its own expected discounted sum of rewards.

A partially observable Markov Game can be described with the tuple: $\langle N, S, A, R, P, O, \gamma \rangle$ in which $O = \{O_1, O_2, ...O_N\}$ which is a collection of observations for each individual agent. The difference is that each agent i cannot acquire a full description of the system state S, but instead obtains a partial observation O_i based on which it selects an action.

NEED TO FINISH

Cooperative vs Competitive

To appropriately maneuver through the landscape of multi-agent reinforcement learning, algorithms will be grouped based on common characteristics and the research will focus on the algorithms most suited for this research. The first categorization is often made in literature on multi-agent systems and labels a system as either cooperative or competitive [33].

Agents pursue a common goal in a cooperative environment. They assist each other to maximize a global reward. In such systems, no limit exist on the amount of system knowledge that one agent can have. Agents can expect that the observed behaviour from other agents originates from good intentions. Agents in a multi-agent system in a competitive setting aim to maximize their own reward.

Thus, the essential difference between cooperative and competitive system is that in a cooperative systems agents maximize a group utility and in competitive systems agents strive to maximize their own utility. In this research, a system is labelled based on their design intention and not their actual behaviour. It could for example be the case that a system that is designed in a competitive manner, exhibits cooperative behaviour amongst agents. This occurs when collaboration translates to the highest individual utility which was the initial goal of the selfish agent.

From the perspective of this literature review, multi-agent systems are classified as cooperative. The objective of the research is to improve an existing CD&R algorithm in a multi-agent setting. All aircraft have common goals which will be expressed in the reward function in the RL algorithm. Those goals have not been set yet, but a very likely goal for example is to minimize the total distance travelled by all aircraft. This is obviously a common goal since the optimal solution since it performance is based on all aircraft and not an individual aircraft. The optimal solution might result in sub-optimal flight paths for some aircraft, but that would still result in the optimal global solution. The performance metrics of the developed algorithm will be based on all agents. Further sections will assume a cooperative setting unless specifically mentioned otherwise.

2.3.3. Challenges in Multi-Agent Reinforcement Learning

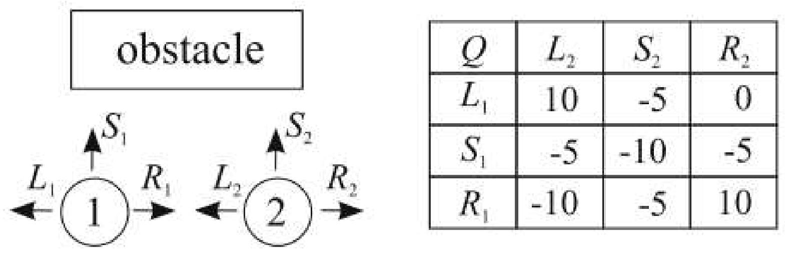
Similar to the single-agent case, proof of convergence for multi-agent tabular methods exists. In [9], an overview of such methods is provided. The discussed methods are based on temporal-difference learning, game theory and direct policy search. The challenges in multi-agent RL learning will be explained with algorithms for which convergence proofs exists. The four challenges which are discussed are nonstationarity, variance in policy-based methods and scalability.

Coordination

Assume a Markov game with N agents which is fully cooperative and the state of the system is fully observable with no bias. All agents have a joint reward function ($R_1 = R_2 = ... = R_N$). If a centralized scheme with a central controller is implemented, the problem reduces to an MDP. The centralized policy takes the full state as input and outputs a joint action for all agents. More on centralized controllers in subsection 2.3.4. A Q-learning algorithm (Equation 2.2.8) can then be implemented to find the optimal Q-function and then use a greedy policy for the optimal policy [9].

However, agents can take actions independently. Therefore, in a scenario in which multiple optimal joint actions are available, coordination between agents is required to guarantee that the individual actions of the agent result in an optimal joint action.

The need for coordination is illustrated in Figure 2.23. In this situation, agents 1 and 2 approach an obstacle which they want to avoid. From the Q-table on the right of Figure 2.23, it can be seen that $Q(L_1, L_2) = Q(R_1, R_2) = 10$. Therefore, multiple joint-actions are optimal. Without coordination agent 1 could select action R_1 and agent 2 could select action L_2. The Q-value for $Q(R_1, L2) = -10$ which is obviously a suboptimal action.



Q	L_2	S_2	R_2
L_1	10	-5	0
S_1	-5	-10	-5
R_1	-10	-5	10

Figure 2.23: On the left a situation with two agents is depicted which want to pass an obstacle and require coordinate to successfully accomplish this task. The Q-table for this situation is visualized on the right. Adapted from. [8]

Nonstationarity
Similar to the previous sections, the Q-learning algorithm will also be used to illustrate the challenge of non-stationarity in multi-agent environments. Q-learning can be implemented in a multi-agent setting by enabling each agent i to learn a state-value function Q_i and assuming that the other agents belong to the environment. Because each agent independently updates their Q-function and thus their policy, the environment becomes non-stationary as seen from another agent. $P\left(s' \mid s, a_i, \pi_1, \ldots, \pi_N\right) \neq P\left(s' \mid s, a_i, \pi'_1, \ldots, \pi'_N\right)$ when any $\pi_i \neq \pi_i^{'}$ [54] describes the non-stationarity of the environment which causes violation of the Markov Property and results in a non-convergent algorithm.

Policy-based methods
Previous challenges were explained with aid of the Q-function which is a value-based method. Challenges for policy-based methods, however, also exist. As explained in subsection 2.2.9, one of the main problems of policy based methods is the variance in the gradient estimate. This variance is amplified in a multi-agent setting since a reward in a multi-agent setting is not only dependent on the own agent's action, but also on the actions of the other agents [54]. This dependency increases the noise and variance in the reward for the single agent and thus increases the variance in the gradient estimate. An increase in the variance of the gradient estimate reduces the probability that the parameters of the policy are updated in the direction of the optimal policy [64].

Scalability
Start with the same scenario sketched in the section on coordination, a centralized policy which determines the joint action for agents. Another consequence of a centralized policy is that each additional agent adds a dimension to the observation and action space, causing the computational complexity of the Q-learning algorithm to exponentially increase [9]. This is often referred to as the curse of dimensionality and requires a large Q-table.

A major issue with RL methods which require tabular storage of Q-values or policies is the scalability. When the size of the state and action space increase or even becomes continuous, these methods become impractical or even infeasible. Therefore, similar to the single-agent case, research on the multi-agent setting will focus on methods with function approximators which are able to map the state space to the action space in a more computationally compact manner [9]. More specifically, the function approximators will be deep ANN. Deep multi-agent reinforcement learning denotes the application of a deep function approximator in a multi-agent reinforcement learning setting. Therefore, the concepts and methods in the subsequent sections are explained in the context of deep multi-agent RL. The proposed methods will proof their value in a more empirical manner similar to the DQN for the Atari games [3].

2.3.4. Control Scheme
The various training approaches for cooperative multi-agent settings are explained in this section. The controller types that are explained are: centralized, decentralized with individual policy and decentralized with shared policy.

Centralized Controller
The original implementation of a centralized controller were the Joint Action Learners (JALs). The name JAL is pretty self explanatory, a central controller decides on a joint action for all agents. The concept has already been introduced in subsection 2.3.3. A central contoller solves the problem of non-stationarity of the environment, but the curse of dimensionality makes a fully centralized approach unfeasible for systems with many agents. Besides, a centralized policy requires continuous communication between the agent and the policy server which is not desired for numerous reasons. The communication can for example be interrupted causing complete disruption of the system. Besides, a time lag always exists between the observation of the agent and the reception of the action from the centralized policy imposing further constraints on the system. An example of a system which successfully implemented a centralized controller is AlphaStar [95]. Alphastar was built by DeepMind and was able to beat top professional players in StarCraft II.

Decentralized with Individual Policy
The foundation for decentralized control in multi-agent system comes from the concept of independent learners (IL) [89]. In a decentralized system each agent decides which action to take based on their own observation. The policy however can be a shared policy or an individual policy. This section zooms in on the decentralized system with an individual policy. Assume a situation similar to subsection 2.3.3: enable each agent i to learn a state-value function Q_i and assume that the other agents belong to the environment. The resulting nonstationary setting provides an additional problem when using a deep Q-function approximator. Namely, the experience replay buffer can not be utilized due to this non-stationarity in the environment [22]. The decentralized control with individual policies does not efficiently scale to a large number of agents since the agents do not share experience with each other which adds to the sample complexity [23].

Decentralized with Shared Policy
In a setting with homogeneous agents, the training process can be ameliorated with parameter sharing. The system is still decentralized (agents take actions based on their own observations), but the policy from which the action is sampled is shared among all agents. Policy parameter updates occur based on a batch of experiences sampled by all agents in the system. The paradigm: "Centralized learning with decentralized execution" is applicable here. With the framework of parameter sharing, single agent methods such as DDPG [77] can be extended to a multi-agent setting [23].

2.3.5. Reinforcement Learning in Multi-Agent Collision Avoidance Setting

The basic concepts of RL were explained in section 2.2 while the function approximators and the challenges of MARL were explained in subsection 2.3.1 and subsection 2.3.3. This section focuses on related research in the multi-agent collision avoidance setting which implement advanced algorithms within the specified domain of value- and policy based methods. Also, an important aspect is to investigate how the algorithms deal with the challenges of MARL. The focus is on the used type of RL model and control scheme. The discrepancy between most existing literature and the goal of this research is that existing research employed RL to completely perform conflict resolution. In this research conflict prioritization is used to improve on an existing CD&R method.

In [96], reinforcement learning was applied to establish a multi-UAV collision avoidance system. A centralized learning, decentralized execution paradigm was adopted in this research. The paper assumes all agents follow the same collision avoidance policy which enables a cooperative decentralized collision avoidance system without communication. To put a constraint on the state and action space, only the 5 nearest neighbours are considered in the collision avoidance process. A two-stage training method is applied. In the first stage the actor policy is pre-trained by trying to mimic the actions of ORCA [94] and the critic is updated according to the standard DDPG algorithm. Thereafter, the algorithm is trained in an unsupervised fashion. Pre-training the algorithm on ORCA resulted in a significantly faster converging algorithm. The final trained policy showed various advantages over the regular ORCA algorithm.

A different approach of implementing reinforcement learning in aircraft collision avoidance was taken in [72]. Instead of designing a model which outputs an avoidance maneuver (e.g. heading or velocity alteration), reinforcement learning was applied to transform a constant parameter called the fixed lookahead time in the Modified Voltage Potential (MVP) algorithm into a dynamic parameter which varies based on relative position of other agents and traffic density. One DDPG policy was trained based on the experience of all agents. The research showed that RL can improve on an existing geometrical method (MVP). A remaining issue however is the stabilization of experience replay. Agents continuously evolve and thus their behaviour keeps changing.

To deal with a varying number of intruding aircraft which should be considered for conflict resolution, [6] introduces a deep multi-agent RL framework with an attention mechanism. The attention mechanism is an LSTM network. One of the main characteristics of an LSTM network is its ability to handle arbitrary-length input sequences. The states of the agents are ordered based on distance and fed to the LSTM network which is able to compress the relevant information of all conflicting agents to a fixed-length vector. Furthermore, a PPO algorithm with centralized training and decentralized execution is utilized.

Comparable scenarios for multi-agent settings which did not include aircraft showed a similar approach in terms of training scheme and utilized model structure. Multi-agent collision avoidance with RL in a robotic setting was proposed by [20]. This research deployed the robust PPO algorithm with centralized learning and decentralized execution. The policy and state-value function are updated independently and share no parameters. The advantage function estimates are computed with GAE [76]. It was found that the designed method significantly outperformed state-of-the-art algorithms in generalization performance, success rate and navigation efficiency. Furthermore, the PPO algorithm is implemented for multiple autonomous vehicles at a non-signalized crossing [92].

2.3.6. Advanced Algorithms

The two model structures that were used were the single-agent Proximal Policy Optimization (PPO) and Deep Deterministic Policy Gradient (DDPG) extended to the multi-agent environment by implementing shared policy among agents. To make a well considered choice for the method, the specifications of the algorithms will first be explained. Implementation of the PPO are often based on an actor-critic approach. Therefore, first the concept of actor-critic methods is explained before the PPO method is elaborated upon. Similarly, DDPG is based on DPG and a description of both is provided.

Actor-Critic Methods

A group of algorithms similar to the algorithms of Equation 2.2.8 and Equation 2.2.8, which take both s_t and the transitioned state s_{t+1} into account when updating their value function, exist in the domain of policy gradient methods. Those algorithms use the one-step return which is equal to the reward r_{t+1} plus the state-value function for the transitioned state $v^{\pi}(s_{t+1})$. If the state-value function is used to assess the quality of action a in a similar fashion, it is called a *critic*. The policy gradients methods which implement such a critic are called *actor-critic* methods. The *actor* refers to the policy which is updated according to Equation 2.35. Since the gradient is partly determined by the critic, the critic aids in the policy update. Implementing an actor-critic method enables online and incremental policy updates since no full episodes are required before the parameters can be updated [88].

A well-known improvement on the regular concept of actor-critic methods is the Asynchronous Advantage Critic (A3C) [57] method. It strives to enhance the stability of the learning algorithm for the on-policy actor-critic method by running multiple instances of the environment running in which multiple agents are asynchronously executed in parallel. This enhances the stationarity of the environment which is a similar goal that the replay buffer in experience replay has. A variation on the A3C is the Advantage Actor Critic or A2C. The first A for asynchronous is omitted, because A2C uses a coordinator to synchronously update the global parameters for all agents. The difference is visualized in **??** It is shown that the A2C version is computationally advantageous compared on a single-GPU compared to A3C [99].

Proximal Policy Optimization

The idea is to limit the size of the policy update to prevent destructing policies. If you keep running gradient descent on the same batch of data, the policy is repeatedly updated and continues to move increasingly further away from the old policy. This also means moving further away from where the batch data was sampled with an increased uncertainty in the loss function characteristics on the area where the new policy operates. This is illustrated with an example in Figure 2.24. The blue arrow represents the update step and at first the update is in the right direction (follows the path), but because the step size is too large, it still results in an undesired situation (fall of cliff).

Figure 2.24: A step size that is too large results in a fall of the cliff. [69]

In the Trust Region Policy Optimatization (TRPO) algorithm constraints were set on the objective function to control the update step size of the policy based. The constraints were based on the Kullback-Leiber (KL) divergence over states [78]. The KL divergence is a measure of how different two probability distribution are. In the original paper of PPO [77] two variations on TRPO were considered. One variation focussed on putting a constraint on the objective function based on an adapative KL penalty. The other method focussed on clipping of the objective function which is arguably simpler and is most widely implemented and showed the best results on the test set. Therefore, the rest of the section explains how the PPO algorithm with a clipped objective function works. The main part of the objective function $L^{CLIP}(\theta)$ can be found in Equation 2.49.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right) \right] \tag{2.49}$$

In which ϵ is a hyperparameter with typical values ranging from 0.1 to 0.3 and $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{Old}}}(a_t|s_t)}$. The first part of Equation 2.49 is the loss function of the conservative policy iteration [77]. The term $\text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right) \hat{A}_t$ performs a clipping operation. Values outside the interval $[1-\epsilon, 1+\epsilon]$ are clipped to the interval corner values. Performing a minimization operation on both terms results in Figure 2.25 for negative and positive advantage values A for a single term t.

The logic behind Equation 2.49 will be explained while keeping the goal of limiting the policy update step in mind. In case of a positive advantage function ($A > 0$) the action is better than following the current policy and thus the probability for this action should be increased in the new policy ($r > 1$). In the left graph of Figure 2.25, belonging to a positive advantage, it can be seen that the update for r is limited to $(1+\epsilon)$. In a similar fashion, for a negative advantage, the reduction in probability of the action is limited by clipping r to $(1-\epsilon)$.

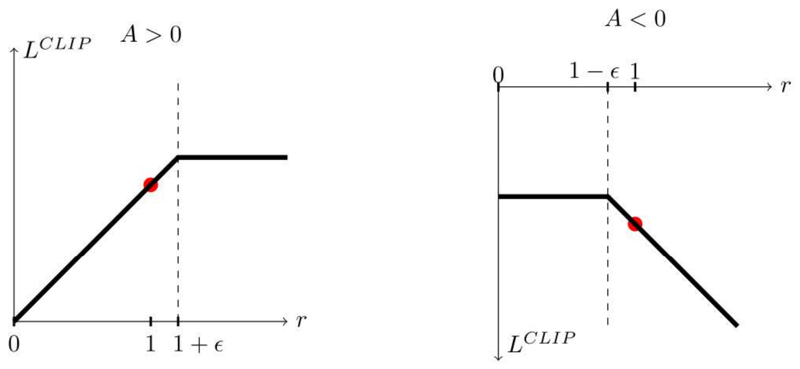


Figure 2.25: Clipped surrogate objective from [77]

Exploration is guaranteed by the stochastic policy. The policy will be updated based on the rewards it receives from the environment with the initial policy. The policy is therefore prone to end up in a local maxima and is highly dependent on the initial policy. Well-known implementations of the PPO algorithm [59] [28] implement KL-divergence as a measurement for early stopping. A small KL-divergence means that the policies did not significantly change between updates and training can be stopped.

Deterministic Policy Gradient
Determinstic Policy Gradient (DPG) has a deterministic policy $\mu_\theta(s)$ which maps states s to actions a. A deterministic policy in the context of policy gradient might sound counter intuitive since there is only one action. However, in [84], proof exists for the Deterministic Policy Gradient Theorem. DDPG is an off-policy actor-critic method. Off-policy DPG methods have a target policy which is different than the behavioural policy.

Deep Deterministic Policy Gradient
The Deep Deterministic Policy Gradient method or DDPG can be seen as an extension of DQN to continuous action spaces [51]. Applying Q-learning to the continuous action space is unpractical with large function approximators since at every time step an optimization of the greedy policy has to be run to find a_t. DDPG is an off-policy actor-critic method based on Deterministic Policy Gradient (DPG) [84].

Similar to the DQN network, a replay buffer is implemented to cope with experience which is gathered in a sequential manner. Futhermore, also the the target network from the DQN network [36] was implemented in the DDPG network in a slightly adjusted fashion for actor-critic methods. The parameters of the target network are updated with a soft update: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$ with $\tau \ll 1$.

To ensure that the values of the features from the observation vector are in a similar range throughout all environments and training phases, the DDPG method implements a paradigm from deep learning called batch normalization [40]. Batch normalization normalizes every input feature across a batch of samples to unit mean and variance.

Since the policy in DDPG is deterministic, the method struggles to provide sufficient exploration in the continuous action space. This problem is tackled by the introduction of a new policy μ' to which noise from noisy process $\mathcal{N}$ is added. Equation 2.50 displays the policy μ' in a more mathematical manner.

$$\mu'(s_t) = \mu\left(s_t \mid \theta_t^\mu\right) + \mathcal{N} \tag{2.50}$$

2.3.7. Discussion on Relevant Literature

The methods in the described relevant literature [96, 72, 6, 20, 92] in subsection 2.3.5 employ a decentralized control scheme with a shared policy. This approach allows for easy scalability to a large number of agents. The centralized critic reduces the variance, increases stability while it still allows for coordination between agents. Therefore, it is advised to follow the approach of centralized learning and decentralized execution in this research when training in a multi-agent scenario.

The model structures that were taken into consideration and for which an extensive description was provided were the PPO and DDPG methods. For the experimental phase of this research, the PPO algorithm is preferred over the DDPG algorithm due to the following:

- **Robustness**: Due to the design of the PPO architecture, the algorithm is inherently more robust compared to the DDPG algorithm and does not rely on experience replay. Even though DDPG has shown to converge in multi-agents settings, it remains unstable. In [66], the PPO algorithm converged on all 4 multi-agent environments and showed state-of-the art performance with limited hyperparameter tuning while DDPG was unable to convergence on any of the environments.
- **Experience Replay**: PPO does not rely on experience replay for convergence
- **Hyperparameters**: Tuning of the hyperparameters of a RL algorithm is a time-consuming task and of utmost importance; slight adjustments in the hyperparameter could transform a divergent algorithm into a convergent algorithm. The PPO algorithm is more robust for hyperparameter settings [66].

- **Action Space**: In this research, a conflict prioritization method is designed and the expected output of the RL algorithm will be binary. As seen from one agent, the algorithm will decide which conflicts to consider (1) and which conflicts to neglect (0). As mentioned in Figure 2.3.6, DDPG can be seen as an extension of DQN to the continuous domain. Therefore, it does not make sense to revert the distinct characteristics of the DDPG network. On the other side, performance of a PPO algorithm was found to increase when switching from a continuous action space to a discrete action space [34].

2.3.8. Description on PPO implementation

The description of the precise characteristics of the PPO method which will be implemented in chapter 3 is given in this section. The PPO agent in this preliminary analysis is a predefined agent from the stable baselines project [28]. The agent is implemented with the PyTorch framework [67]. The implementation in the stable baseline agents follows [77]. PyTorch facilitates automatic differentiating. Therefore, a scalar loss function value is sufficient to update the model parameters. The total loss function is composed of several composed and is defined by Equation 2.51 [77].

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S\left[\pi_\theta\right](s_t) \right] \tag{2.51}$$

In Equation 2.51, $L_t^{CLIP}(\theta)$ is the clipped objective function from Equation 2.49 ,c_1 and c_2 are coefficients belonging to the squared error loss $\left(V_\theta(s_t) - V_t^{\text{targ}}\right)^2$ and entropy bonus S respectively. The entropy bonus encourages explorations as was found by [57].

The state-value function $V(s)$ is used in the computation of the advantage function. Inclusion of the loss due to the error in the estimated state-value function is compulsory when the actor and critic network share parameters. Parameter sharing can for example be useful when the network takes an image as an input and a similar feature extraction mechanism can be used for both the actor and critic network. However, in this research the input is not an image and no shared layers are defined. In theory, this means that c_1 can be set to zero and the parameters of the value network can be optimized separately from the policy parameters. The stable baselines implementation however does not allow this since all parameters are updated based on the scalar loss value computed with Equation 2.51.

The advantage function A_t is defined by Equation 2.52 in which δ_t is defined by Equation 2.53 [77]. Equation 2.52 is a trunctated version of the GAE [76] and based on the implementation of the GAE in the paper on the A3C architecture [57].

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \tag{2.52}$$

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{2.53}$$

3 Preliminary Analysis

To develop a better insight in how to optimally utilize the theoretical concepts defined in chapter 2 to fulfill the research objective, preliminary experiments were performed. The research scope of the preliminary experiments is further specified in section 3.1. A short recap of the methodology is given in section 3.2. Subsequently, the single- and multi-agent experiments are described in section 3.3 and section 3.4 respectively. Finally, the proposal for the research approach in the main part of the thesis is provided in section 3.5.

3.1. Research Scope

In this preliminary analysis, it is decided to not extensively research and fine-tune the optimal network architecture, hyperparameters and the selected features. The aforementioned model properties are greatly dependant on the given task to which the RL algorithm is applied. Elaborate tuning on the proposed scenarios in the single- and multi-agent setting is not very sensible since the final task (in the thesis) will differ from the scenarios which provide a Proof-of-Concept.

The single-agent case and multi-agent case each consider one specific scenario and the generalization property of the model is thus not evaluated. Evaluating generalization performance of a reinforcement learning model is harder compared to supervised learning problems. In a supervised learning problem the total data set can be split in a train, validation and test set. The model is trained with the train and validation data. The generalization abilities are determined based on the performance on the test set. In reinforcement learning, such a division cannot be made. Testing of the generalization abilities would require extensive training on experiments which strive to vary its conditions such that the agents encounters all relevant dynamics to train its model on. This is not within the scope of the preliminary thesis.

3.2. Research Methodology

The research methodology for the experiments has already been described in detail in section 1.2. The Figure 1.1 is inserted below as Figure 3.1 for a quick recap of the methodology in the preliminary phase. Based on literature, the model settings for the single-agent experiment are initialized. Once the results for single-agent experiment are determined to be sufficient, the multi-agent experiment is initialized with results from the single-agent experiment and additional concepts from literature. With a successful multi-agent scenario, the outcome of both experiments is combined to devise a proposal for the design of the the multi-agent experiment in the final part of the thesis.

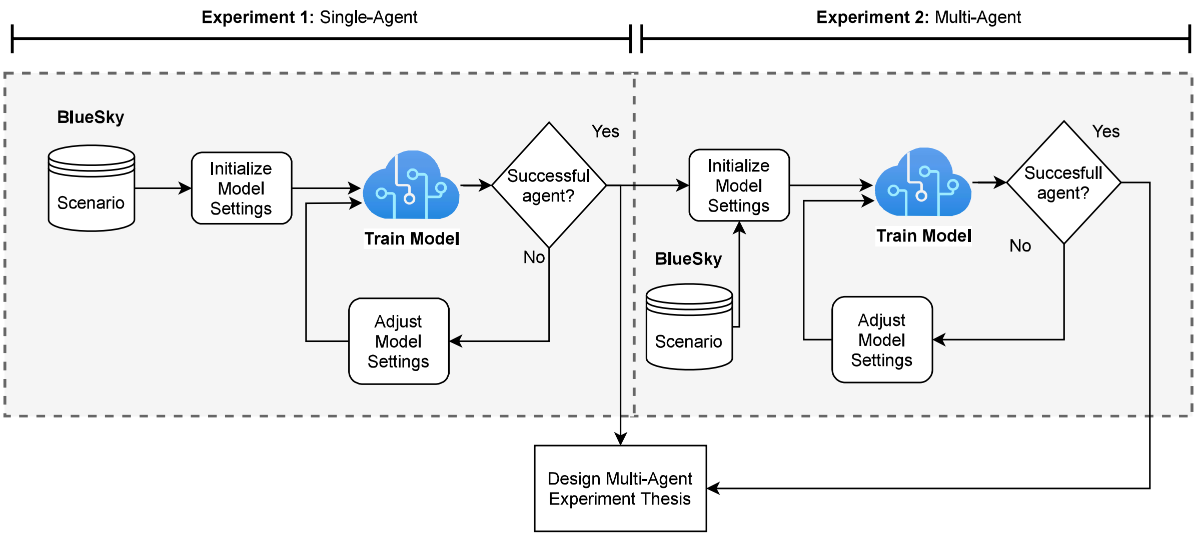


Figure 3.1: Schematic overview of single- and multi-agent experiment and their relation to each other and the design of the multi-agent experiment in the final thesis

3.3. Single-Agent Setting

This section on the experiment in the single-agent setting starts with stating the experiment goals and the experiment scenario in subsection 3.3.1 and subsection 3.3.2 respectively. Thereafter, information on the model structure and the hyperparameter settings are provided in subsection 3.3.3 and subsection 3.3.4. Hypotheses are given in subsection 3.3.5 and the results are displayed in subsection 3.3.6.

3.3.1. Experiment Goals

Besides the main goal of providing a first Proof-of-Concept, several sub-goals of this experiment can be defined:

1. Development of an intuition for the hyperparameters.
2. Get insight in the feature design process. Investigate what the characteristics of a good feature are.
3. Conclusions from single-agent experiment can assist in the initialization of multi-agent scenario.

3.3.2. Experiment Scenario

The name of the single-agent experiment indicates that the SSD algorithm is only enabled in only one aircraft. The SSD is thus disabled in all other aircraft. Empowering all aircraft to use the SSD for conflict resolution results in complex system dynamics in a scenario with high-traffic density preventing straightforward testing of scenarios. The experimental scenario is visualized in Figure 3.2.

In this experiment the Boeing-747 is chosen as aircraft type. The first notable element of the scenario in Figure 3.2 is the number of aircraft. With this high traffic density scenario, the limitations of the current implementation of the SSD algorithm can be illustrated. "AC00" is the controlled aircraft in which the SSD is enabled. All other aircraft fly in a straight line in the direction of their initial heading. The SSD after the scenario is run for 5 seconds is displayed by Figure 3.3. Note that at subsequent figures of an SSD, the legend has been omitted, but is equal to the legend of Figure 3.3. The velocity solution point $V_{solution}$ is computed with the shortest path out rule from Table 2.4, because it was found that this rule was most optimal [2]. Immediately, it becomes evident that the found solution for the the velocity vector is undesired. It requires almost a complete 180° right turn from the current heading. When AC00 is altering its current velocity vector $V_{current}$ to $V_{solution}$, a loss of separation between AC00 and AC04 occurs. Further specifications of the simulated aircraft can be found in Table 3.1.

Table 3.1: Characteristics of Simulation Aircraft in BlueSky

Aircraft Type	Altitude	Velocity
Boeing 747-400	FL250	250 kts

The aim of this experiment is for the RL algorithm to activate/deactivate the VOs of the aircraft such that no loss of separation occurs for AC00 while limiting the deviation from the original flight path. The solution of the SSD is still computed with the shortest path out rule. AC00 reaches its target waypoint if the initial trajectory was propagated and AC00 would fly in a straight line with heading 0.

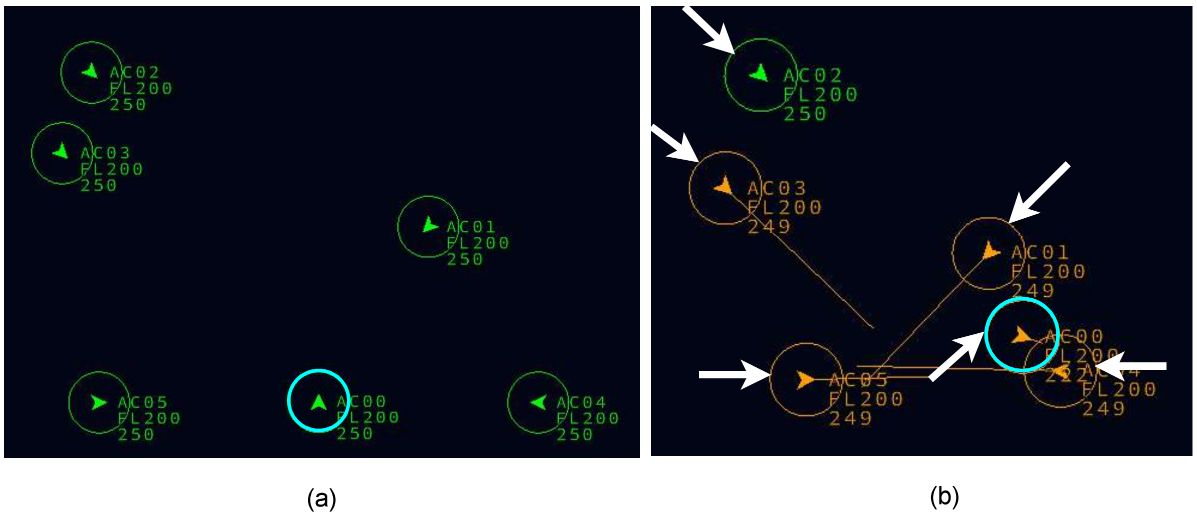


Figure 3.2: Screen capture of the BlueSky Simulator which displays the experiment scenario with the original implementation of the SSD algorithm. The two screen captures represent the initial state (a) and the LoS of AC00. The blue circle indicates aircraft AC00

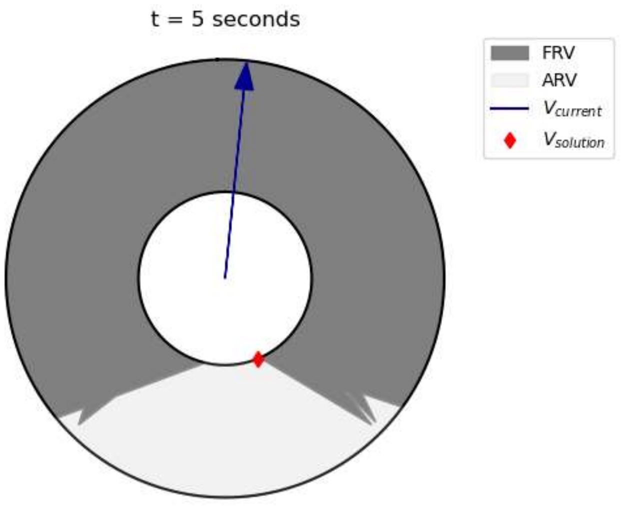


Figure 3.3: SSD of base case after 5 seconds

3.3.3. Model

In subsection 2.3.6, the PPO algorithm was selected for this research. The implementation of the PPO algorithm in the stable baselines project was also described [28]. The relation between the collecting of experience and updating of the model weights is explained in this section. Understanding how gathered experience and model updates interact is important to comprehend the hyperparameters of the model and tune them if necessary. Thereafter, the architecture of the actor-critic network is described.

Updating Model Weights
The weights θ are updated according to the pseudo-code of algorithm 1. Each iteration, N parallel actors gather experiences from their environments and compute advantage estimates for T timesteps. The parameters are updated with an optimizer for K epochs on a minibatch sampled from the NT time step data points. Parallel actors allow for faster collection of experiences, in this preliminary analysis only one environment and thus one actor was utilized [77].

Algorithm 1: Proximal Policy Optimization (PPO) [77]

```
**for** *iteration=1,2,...* **do**
    **for** *actor=1,2,...,N* **do**
        Run policy $\pi_{\theta_{old}}$ in environment for T timesteps
        Compute advantage estimates $\hat{A}_1,...,\hat{A}_T$
    **end**
    Optimize surrogate $L$ wrt $\theta$, with $K$ epochs and minibatch size $M \leq NT$
    $\theta_{old} \leftarrow \theta$
**end**
```

In order to start with the tuning of hyperparameters or understand why the default hyperparameters settings make sense, it important to first properly understand what the precise definition of an epoch, batch and mini-batch is and how they are related to each other. A batch consists of NT data points and represents the entire available training data set for one iteration. A mini-batch is a subset of the batch data. It is advised to select the size of the mini-batches such that fits an integer times in the batch. Otherwise, the last parameter update of the batch will be performed on a mini-batch which size is smaller than the specified mini-batch size, because no sufficient amount of data is left to construct an entire mini-batch.

Parameter updates can be based on an entire batch which is computationally slow and intractable for large datasets from a memory perspective. Parameter updates can also be updated for every sample. This results in high variance in the parameter updates which can be advantageous to escape poor local minima, but makes convergence to the desired minima more complicated since it could also escape the desired minima. Therefore, performing weight updates based on a mini-batch is a trade-off between the two aforementioned mentions and profits from the advantages of both methods [75].

An epoch indicates how often all the data from a batch is utilized for model parameter updates. So one epoch means that the entire batch of training data is used once for a parameter update. Advantage normalization is applied per mini-batch according to Equation 3.1 in which $\sigma_{\hat{A}}$ and $\mu_{\hat{A}}$ represent the standard deviation and mean of the mini-batch of advantage estimates respectively.

$$\hat{A}_{i_{norm}} = \frac{\hat{A}_i - \sigma_{\hat{A}}}{\mu_A} \tag{3.1}$$

Architecture Actor-Critic Network
The architecture of the Actor-Critic Network is visualized in Figure 3.4. The actor denotes the stochastic policy $\pi_\theta(s)$ and the critic the value function $V_\theta(s)$. The actor network takes an observation as input and outputs a probability distribution over the discrete actions from which an action is sampled which is referred to as a categorical policy. The critic network takes the same observation as input and outputs a scalar value for the state-value function. The actor and critic network do not share any layers, both have two hidden layers with 64 neurons per hidden layer and a tanh activation function. Orthogonal initialization is utilized for initialization of layer weights.

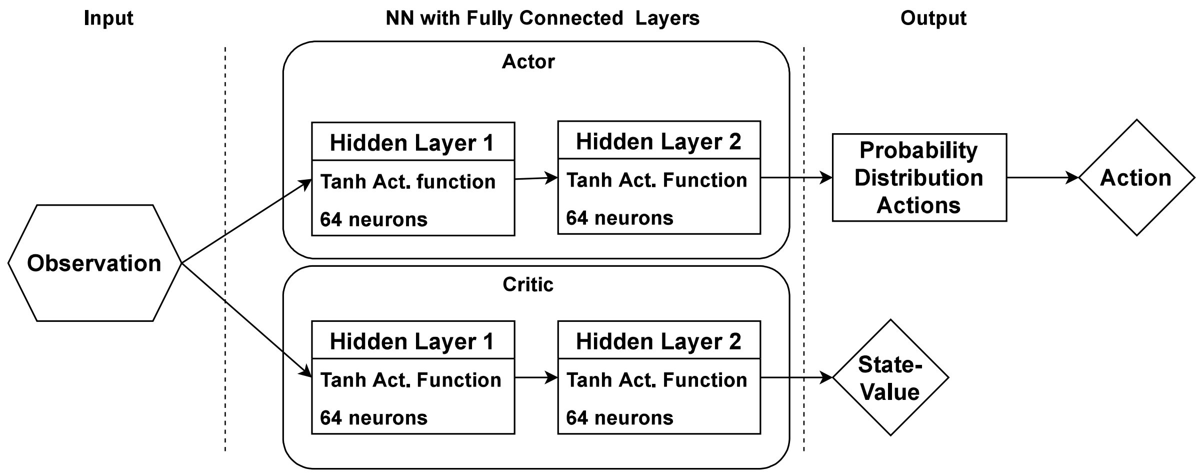


Figure 3.4: Architecture of the Actor-Critic network for the single-agent experiment

3.3.4. Model Settings

Environment
The RL environment is synthesized with the standardized framework of OpenAI Gym [7]. In an OpenAI Gym environment three functions have to be defined. An initialization function in which the dimensions of the observation and action space are defined. A step function which allows the agent to take a step in the environment based on the inputted action. The step function takes an action as input and returns an observation and reward. Finally, the reset function resets the environment when the episode ends or a terminal state is encountered. The episode ends if the specified maximum episode length is reached.

Observation Space
In this environment, the observation space is a discrete representation of the continuous observation space. The size of the observation space vector is defined by multiplying the number of aircraft that are taken into account with the number of features per aircraft. Deciding how many aircraft and which aircraft should be taken into account for conflict avoidance will be an important design decision in the final model. In this scenario, the observation space of AC00 collects observations from all 5 other aircraft during the entire experiment.

The features selected for this experiment are specifically chosen for this scenario. A feature denotes a property which can take on a numerical value and can be assigned to an aircraft. Relevant features in the context of applying conflict prioritization to conflict resolution can for example be t_{cpa} and d_{cpa} between ownship and intruder. Determining which features are optimal for conflict prioritization is a critical task and is not an exact science. Features from Table 2.3 are a good starting point. The multi-agent scenario restricts the number of features which can be selected since every feature has to be selected from 5 aircraft effectively multiplying the observation space dimension with 5.

Incorporating the minimum number of features has the advantage that the size of the observation space remains limited. Besides, it is interesting to investigate the power of the agent with a limited observation space. The distance d between AC00 and the other aircraft was chosen as the first feature, because this could allow the agent to prioritize aircraft based on relative distance and potentially learn that aircraft with a smaller d probably form a greater threat than aircraft with a large d. The second feature is the time to closest point of approach t_{cpa}. It is hypothesized that the agent can combine both features to correctly judge on which aircraft the conflict resolution maneuver should be based. The vector of observations $\mathbf{O}_t$ is defined by Equation 3.2.

$$\mathbf{O}_t = \{t_{1_{cpa_t}}, d_{1_t}, t_{2_{cpa_t}}, d_{2_t}, t_{3_{cpa_t}}, d_{3_t}, t_{4_{cpa_t}}, d_{4_t}, t_{5_{cpa_t}}, d_{5_t}\} \tag{3.2}$$

An overview of the feature specifications can be found in Table 3.2 in which N_{points} indicates the number of discrete points on the continuous range. Both the range and number of discretization points are additional hyperparameters. The lenght of the observation vector is thus 10 (five aircraft and 2 features per aircraft) and each observation can take on 15 discrete values.

Table 3.2: Specifications Observation Space

Feature	Range	N$_{points}$	Unit
t_{cpa}	[0,60]	15	s
d	[0,120]	15	km

Action Space

The action space should be able to turn VOs on (1) and off (0). It therefore needs an output for all selected aircraft. Again, in this scenario all 5 other aircraft. For each aircraft i action $a_i \in \{0,1\}$. The complete action space A at any given time for AC00 can be found in Equation 3.4 and the action vector at time t $\mathbf{a}_t$ in Equation 3.4.

$$A = \{\{0,1\},\{0,1\},\{0,1\},\{0,1\},\{0,1\}\} \tag{3.3}$$

$$\mathbf{a}_t = \{a_{1_t}, a_{2_t}, a_{3_t}, a_{4_t}, a_{5_t}\} \tag{3.4}$$

Reward Function

As stated in subsection 3.3.2, the two objectives of this research are to prevent loss of separation for AC00 while minimizing deviation from target waypoint. The reward function should incorporate both these elements. The reward function can be found in Equation 3.5 in which d_g denotes the distance between AC00 and its target waypoint at time t in km. The term $d_{g_{init}}$ represents the distance at $t = 0$.

$$r(s_t) = \begin{cases} -100, & \text{Loss of Separation, end episode} \\ -\frac{d_g}{d_{g_{init}}}, & \text{otherwise} \end{cases} \tag{3.5}$$

Loss of separation is a sparse reward. The rewards with respect to the distance from target waypoint help the algorithm converge and optimize its efficiency. The reward is negative, because if AC00 moves towards its target waypoint, it will become less negative which can be interpreted as positive by the agent. Trying to minimize number of conflicts by penalizing conflicts in the reward function sounds tempting, but in the MVP it was found that conflicts actually positively contributed to the conflict resolution [31]. The reward function is based on the approach of [102]. If no LoS occurs the episode is stopped after 200 seconds of simulation time.

Hyperparameter Settings

In subsection 2.3.7, it was stated that the PPO algorithm is a robust algorithm and generally does not require extensive tuning of its hyperparameter settings. Therefore, it was decided to first evaluate the performance of the algorithm based hyperparameter values from literature [77]. Based on the initial results, a decision on the required type of hyperparameter optimization was made. The default hyperparameter settings were found to be sufficient to obtain an agent with a well performing behaviour. Manual tuning did not result in an increase in performance. A summary of the hyperparameters of the PPO algorithm can be found in Table 3.3. The max gradient parameter is from gradient clipping. Gradient clipping is implemented to limit the size of the gradient to counteract exploding gradients in a deep neural network.

Table 3.3: PPO algorithm hyperparameter specifications

Parameter	Value
Horizon (T)	2048
Learning rate	$3 \cdot 10^{-4}$
Number epochs	10
Minibatch size	64
Discount (γ)	0.99
GAE parameter (λ)	0.95
c1	0.5
c2	0
Clipping (ϵ)	0.2
Max gradient	0.5

3.3.5. Experiment Hypotheses

The following hypotheses for the single-agent experiment are established:

1. **AC00 can maneuver through the scenario without a LoS**. Aircraft AC00 should initially ignore AC02 and AC03 and focus on avoiding AC01. Ignoring AC02 and AC03 refers to the deactivation of their VOs in the SSD diagram of AC00. Once AC01 is avoided it should reactivate the VOs from AC02 to maneuver through this scenario without any loss of separation.

2. **AC00 maneuvers through the scenario with minimal deviation from optimal flight path**: The VOs should be activated and deactivated in a way which minimizes the increase in flight path length while preventing LoS.

3.3.6. Experiment Results

In this section, first the training progress is visualized. Subsequently, the trajectory of the trained agent in the scenario is displayed. Finally, a discussion on conclusion of the results is given.

Training Progress

The training progress for the total loss $L_t^{CLIP+VF}(\theta)$ and value loss $L_t^{VF}(\theta)$ is displayed by Figure 3.5 and Figure 3.6 respectively. The agent is trained for 70000 steps and each model step corresponds to 1 second in the simulation time of the BlueSky Simulator. The maximum trajectory length is 200 steps if no LoS occurs. Both the total loss function and value function were convergent.

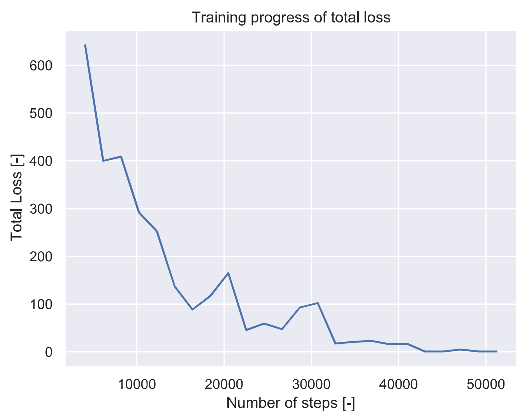


Figure 3.5: Total loss of centralized model for increasing number of steps

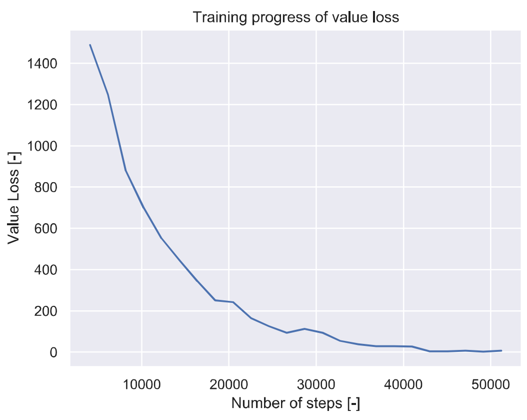


Figure 3.6: Value loss of centralized model for increasing number of steps

Flight Path Trained Agent

The RL model was able to learn the fastest and safest way to resolve this conflict situation by applying conflict prioritization. A visual representation of the optimized path that the trained agent follows can be found in

Figure 3.7. The agent AC00 learned to maneuver through the environment without any loss of separation. To avoid aircraft AC01, agents AC00 has to adjust its flight path and make a slight deviation of the desired heading to the left. The new flight path of AC00 causes a conflict with AC02. Thus, as soon as the conflict with AC01 is resolved, AC00 reactivates the VO of AC02 and avoids AC02 as well. Another notable remark which does not become obvious from Figure 3.7, is that the flight path of aircraft AC00 shows a slight oscillatory motion due to the activation/reactivating of VOs which denotes that the agent aims to minimize the deviation from original flight path. The optimization objectives in the single-agent experiments are assessed in a more qualitative manner. In the final thesis, the performance will be measured in a quantitative way based on the metrics of Table 2.1.8.

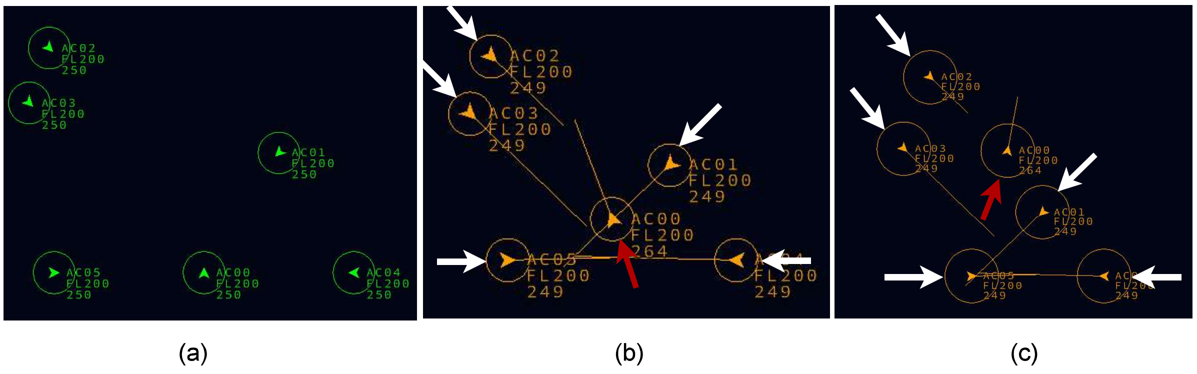


Figure 3.7: Three screen captures of the BlueSky Simulator which show the solution of the agent in the single-agent experiment. The three screen captures represent the initial state (a), conflict avoidance maneuver between AC00 and AC01 (b) and the conflict avoidance maneuver between AC00 and AC02 (c)

In Figure 3.8, the activation/deactivation of the VOs of intruding aircraft for ownship AC00 are visualized for the first 200 seconds of simulation time. The most interesting VOs belong to AC01 and AC02. Initially, the VO of AC01 is mainly activated and the VO of AC02 is completely deactivated. Between 25 and 100 seconds it can be seen that the VOs of AC01 and AC02 are continuously switched on and off to allow AC00 to maneuver through the environment in an efficient manner without LoS.

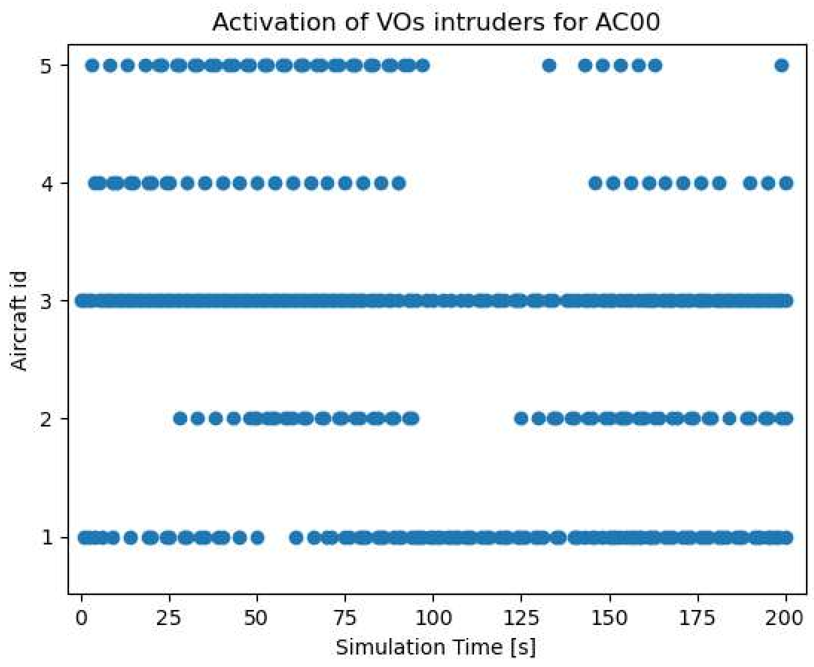


Figure 3.8: Visualization of activation/deactivation of velocity obstacles for intruding aircraft for AC00 in the first 175 seconds of simulation time.

Discussion

The default hyperparameters provide satisfactory performance on the single-agent experiment. In the training architecture for the single-agent experiment, the horizon parameter is equal to the batch size since only one actor collects experience. It was found that the batch size has great influence on convergence. Reducing the horizon parameter before updating the parameters resulted in a bad performing agent. The batch size denotes the number of steps the agent takes before the policy is updated. If the number of steps is not sufficiently large, the agent could get trapped in a local optimum. The agent begins to optimize its policy while it has not interacted enough with the environment to know the most optimal direction to update its weights in. Instead, it settles for a local optimum which is inferior to the global optimum.

The agent was able to successfully maneuver through the scenario without a LoS. The oscillatory motion was an indication that the agent also optimized its solution with respect to efficiency. The agent was able to learn this behaviour based on the feature set $\{t_{cpa},d\}$. It was expected that this feature set would suffice, because if the agent can learn that aircraft with a small t_{cpa} and d impose the greatest threat, the agent should be able to successfully maneuver through this particular scenario.

The reward function eventually functioned as expected, but it required several design iterations and manual tuning of the weights before the final behaviour was achieved. If the -100 reward for a LoS was reduced to for example -5, the agent AC00 learned that it was more advantageous to simply fly in a straight line and not deviate from the flight path than to avoid the loss of separation. Therefore, in the multi-agent case, it is advised to utilize the same reward function. If an alteration of the reward function is necessary, ensure that magnitude of the rewards for loss of separation and deviation from flight path are recalibrated.

Conclusion

The following conclusions were drawn from the single-agent experiment:

- Proof-of-Concept for the single-agent setting is given.
- No hyperparameter tuning was required. Therefore, it is advised to start with the default hyperparameter setting in the multi-agent experiment.

- The feature set $\{t_{cpa}, d\}$ contained sufficient information for the agent to correctly activate/deactivate the VOs of intruding aircraft.
- The reward function in the single-agent setting performed accordingly and serves as a good starting point in the multi-agent setting.

3.4. Multi-Agent Setting

The structure is similar to the single-agent setting. The experiment goals and the experiment scenario are described in subsection 3.4.1 and subsection 3.4.2 respectively. Subsequently, the decision on the control scheme and hyperparameter settings are provided in subsection 3.4.3 and subsection 3.4.4. Hypotheses are described in subsection 3.3.5 and the results can be found in subsection 3.3.6.

3.4.1. Goals Multi-Agent Experiment

The goals of the experiment in the multi-agent setting are:

- Prove that it is possible to show cooperative behaviour between agents in a multi-agent.
- Provide insights into the pitfalls of designing a multi-agent scenario.
- Assist in determining the scope of multi-agent experiment setting.

3.4.2. Experiment Scenario

Designing a base case for the multi-agent setting is significantly harder than in the single-agent setting, because two agents are adapting their velocity vector based on the interactions with the environment inciting more unpredictable system dynamics. The main goal of the experiment is to proof that cooperation between agents is feasible. The experiment should therefore be designed in such a way that the agents must cooperate to come to a good solution.

An overview of the initial state of the experiment in the BlueSky simulator can be found in Figure 3.9. In this experiment, conflict resolution with the SSD is enabled in AC00 and AC01. The other aircraft simply fly in a straight line in the direction of their initial heading. In the current implementation of the SSD, the conflict between AC00 and AC01 cannot be resolved without a LoS for AC01. The aim of this experiment is to investigate whether the aircraft AC00 and AC01 can synthesize a joint solution which exceeds performance of the regular SSD algorithm. The number of LoS (N_{LoS}) is measured and efficiency of the flight path is again assessed in a more qualitative manner.

In a similar scenario with only AC00 and AC01, a geometrically optimal solution exists according to Ellerbroek [16]. However, the construction of the geometrically optimal solution in Figure 3.9 is distorted by AC02, AC03, AC04 and AC05, because the velocity obstacles of belonging to these aircraft significantly reduce the set of ARVs in the SSDs of AC00 and AC01. The solution space of AC00 and AC01 at $t = 0$ can be found in Figure 3.10 and Figure 3.11 respectively.

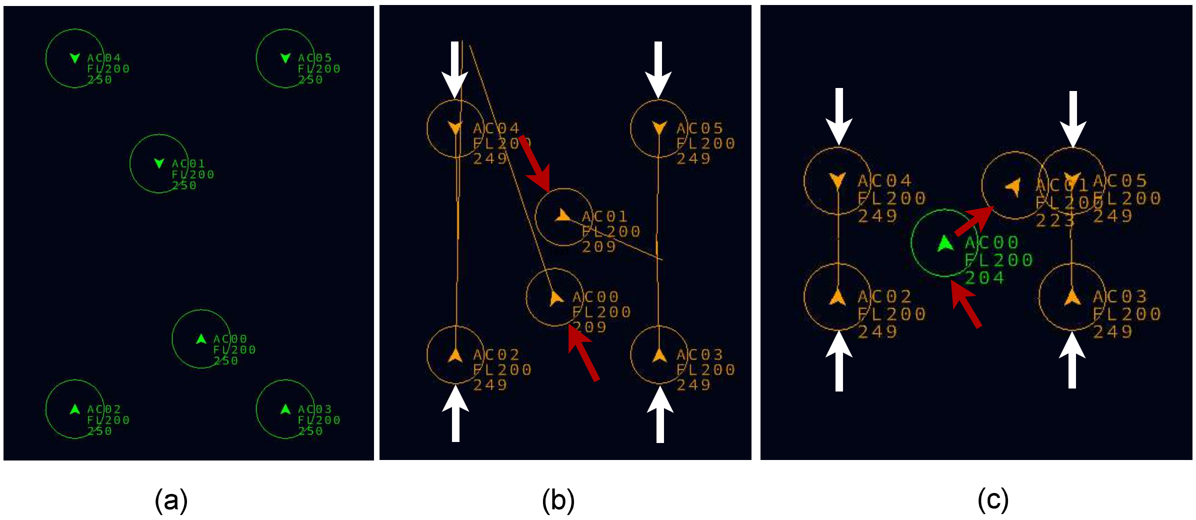


Figure 3.9: Three screen captures of the multi-agent experiment with the original implementation of the SSD algorithm. Conflict resolution with SSD is activated in AC00 and AC01. The three screen captures represent the initial state (a), the start of conflict maneuver of AC00 and AC01 (b) and the loss of separation of AC01 (c).

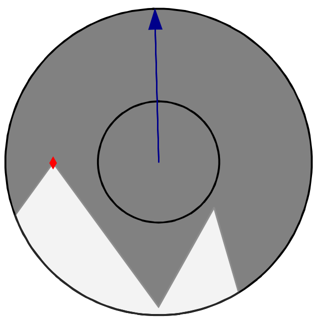


Figure 3.10: SSD at $t = 0$ s for AC00

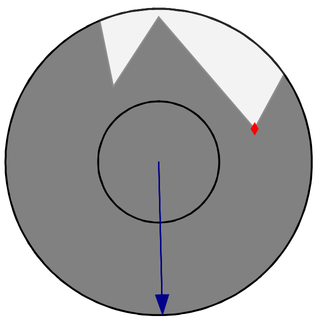


Figure 3.11: SSD at $t = 0$ s for AC01

3.4.3. Model

First, a decision has to be made on which control scheme (subsection 2.3.4) will be used for multi-agent RL. The most natural choice for a control scheme which allows for cooperation between agents and ensures stability is the centralized controller (subsection 2.3.4). The major disadvantage of the centralized controller is that it poorly scales to a large number of agents. Since the number of controlled and observed agents are now only 2 and 5 respectively, the dimensionality of the observation and action space is not expected to be the limiting factor for a successful multi-agent model. Therefore, the centralized controller is deemed suitable for providing a proof-of-concept for the cooperative behaviour of the agents, even if the implementation in the final thesis will follow a decentralized control scheme. If the centralized controller cannot converge to a joint solution, it implies that the decentralized control method will also fail, because it is assumed that attaining convergence for a decentralized controller is more difficult compared to a centralized controller. If The centralized controller does converge, a possibility exists that the decentralized controller also converges. The implementation of the PPO model is similar to the single-agent case of section 3.3.

3.4.4. Model Settings

The observation and action space have to be extended to the multi-agent scenario. Due to the centralized control scheme, the observation and action spaces of the individual agents can be stacked to compose the observation and action space of the centralized model. Consequently, the stacked observation ($\mathbf{a}_c$) and action vector ($\mathbf{a}_c$) of the centralized model at time t can be computed with Equation 3.6 and Equation 3.7 respectively.

$$\mathbf{O}_{c_t} = \{\mathbf{O}_{00_t}, \mathbf{O}_{01_t}\} \tag{3.6}$$

$$\mathbf{a}_{c_t} = \{\mathbf{a}_{00_t}, \mathbf{a}_{01_t}\} \tag{3.7}$$

The action vector length thus doubles to 10 discrete actions since 2 aircraft each decide on activation of 5 VOs. If similar to section 3.3 two features per aircraft are used, the observation input vector is also doubled in size. Addition of features obviously increases the observation vector even further. The reward function from Equation 3.5 is applied for both agents AC00 and AC00. The total reward r_{total} is computed with Equation 3.8.

$$r_{total} = r_{00} + r_{01} \tag{3.8}$$

3.4.5. Experiment Hypotheses

The following hypotheses were formulated with regard to the experiment in the multi-agent setting:

- **AC00 and AC01 should not enter a loss of separation while simultaneously maintaining high efficiency**. Similar to the single-agent experiment, the agents are expected to avoid LoS and optimize their behaviour with respect to efficiency.
- **More steps are required for a converging method compared to the single-agent setting**. It is expected that establishing cooperation between agents is difficult for the PPO algorithm. Thus more experience is required for training.
- **Cooperation can be achieved with a centralized controller**. Due to the centralized control scheme, the multi-agent problem can be solved with a single-agent approach. Convergence was achieved for the experiment in the single-agent setting. Hence, it is expected that AC00 and AC01 can find a solution which is better than the solution described in Figure 3.9.

3.4.6. Experiment Results

Training Progress

The platform on which the models were trained was similar to subsection 3.3.6. A step again corresponds to 1 second in simulation time of the BlueSky simulator. The agents were trained for 30000 steps. If no terminal state is encountered, the episode ends after 200 steps. The training progress for the total loss $L_t^{CLIP+VF}(\theta)$ and value loss $L_t^{VF}(\theta)$ is displayed by Figure 3.12 and Figure 3.13 respectively.

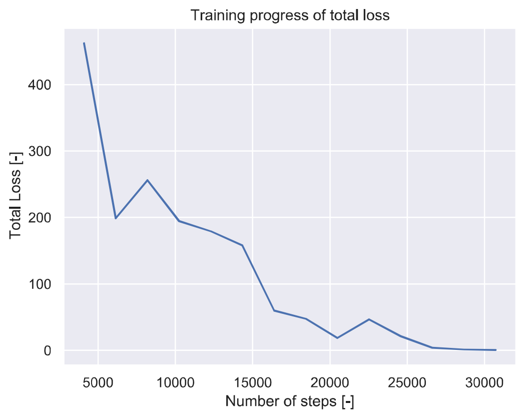


Figure 3.12: Total loss of centralized model for increasing number of steps

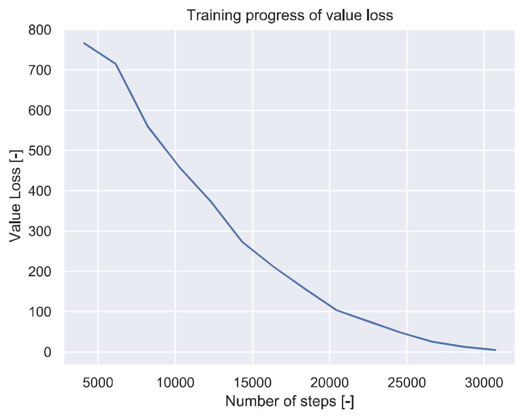
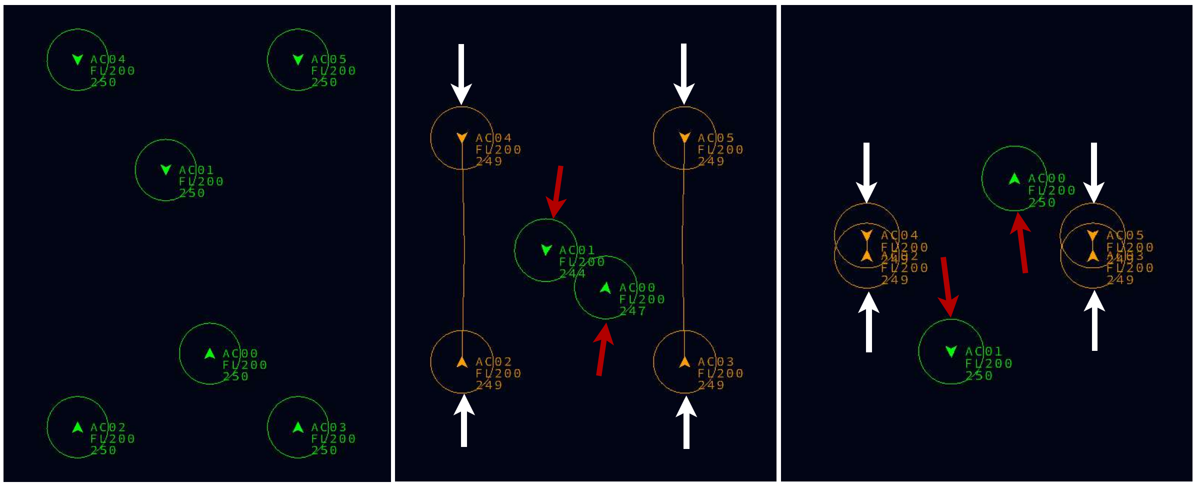


Figure 3.13: Value loss for centralized model for increasing number of steps

Flight Paths Trained Agents

The RL model with a centralized controller enabled the agents to cooperate and find a better solution based on the amounts of LoS and flight path distance. Figure 3.14 displays the flight paths of the aircraft while the RL model decides which VOs AC00 and AC01 have to incorporate in their SSD. It becomes evident that both have succeeded to find a joint solution which is preferred over the solution constructed by the current SSD-based algorithm. No LoS occurs for AC00 or AC01.



(a) (b) (c)

Figure 3.14: Screen capture of the BlueSky Simulator which displays three screen captures the solution for the multi-agent scenario found by the centralized controller. The three screen captures represent the initial state at $t = 0$ s (a), begin conflict avoidance maneuver between AC00 and AC01 at $t = 60$ s (b) and the end of the conflict avoidance maneuver between AC00 and AC01 at $t = 150$ s(c)

The SSD for AC00 and AC01 at $t = 0$ are displayed in Figure 3.15 and Figure 3.16 respectively. As expected, compared to Figure 3.10 and Figure 3.11, VOs in Figure 3.15 and Figure 3.16 have been omitted to allow for a more optimal joint solution.

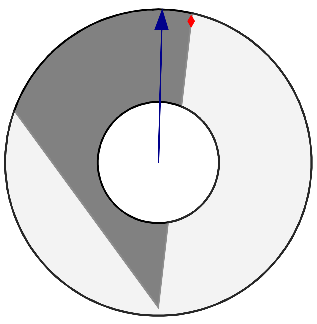


Figure 3.15: SSD at $t = 0$ s for AC00

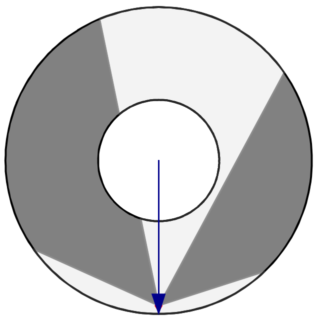


Figure 3.16: SSD at $t = 0$ s for AC01

The activations/deactivations of the velocity obstacles for the intruding aircraft for AC00 and AC01 are visualized in Figure 3.17 and Figure 3.18 respectively. A blue dot indicates that for that simulation step, the VO of a particular aircraft was included in the SSD for conflict resolution. From Figure 3.17, it can be seen that AC01 mainly deactivates the VO of AC00 and AC02 almost always activates the VO of AC04. While AC00 in Figure 3.18 does activate the VO of aircraft AC01, AC02 and AC05 most of the time and the VOs of AC03 and AC04 most of the time.

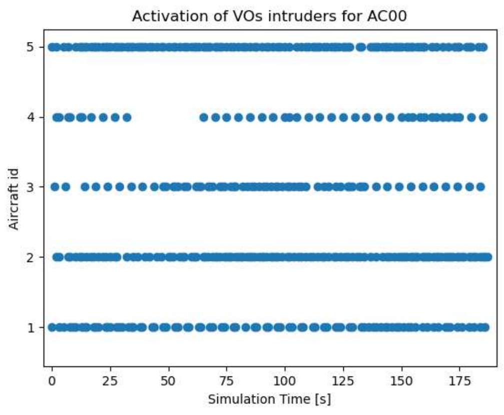


Figure 3.17: Visualization of activation/deactivation of velocity obstacles for intruding aircraft for AC00 in the first 175 seconds of simulation time.

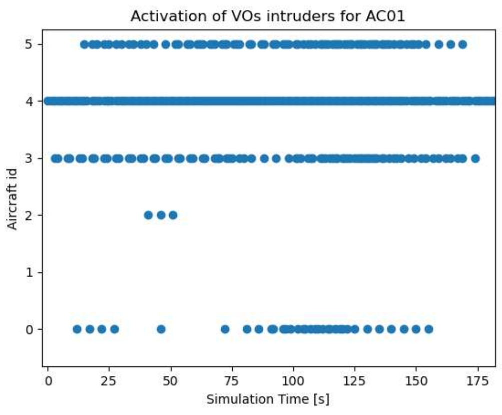


Figure 3.18: Visualization of activation/deactivation of velocity obstacles for intruding aircraft for AC01 in the first 175 seconds of simulation time

Discussion

The aircraft AC00 and AC01 did both not enter a LoS and AC00 and AC01 were able to increase the total efficiency of their combined flight paths. It was hypothesized that the multi-agent scenario required more steps for convergence. However, the multi-agent scenario converged in 30000 steps compared to the single-agent scenario which converged in 70000 steps. The multi-agent model is said to be more sample efficient than the single agent model. This can be explained based on two reasons:

- **Scenario Length**: The most prominent problem in the multi-agent case is the immediate conflict between AC00 and AC01. In the single-agent case the relevant conflicts are with AC01 and AC02 which requires more steps in the environment to sample experience from than the single immediate conflict in the multi-agent case.
- **Scenario Complexity**: The single-agent case is more complex, because the agent first has to learn how to avoid AC01 and subsequently has to learn how to avoid AC02. It is expected that more batches of data are required before converging to the desired behaviour. Once the multi-agent model is able to solve the initial conflict between AC00 and AC01, it is expected that agents AC00 and AC01 will be able to maneuver through the rest of the environment without the need of conflict prioritization.

Conclusion

The following concluding remarks can be made based on the multi-agent experiment:

- The Proof-of-Concept for multi-agent scenario provided. Cooperation between aircraft is feasible in a scenario with a centralized controller.
- The selected features t_{cpa} and d allowed the model to converge in the multi-agent setting. The features need to be altered for the multi-agent case in the final experiment, because with the current set of features the agent do not know the relative position of the other aircraft. The agents were still able to be trained on the current set of features, because relative position was not imperative to solve the single- and multi-agent experiments. In a more generalized setting, relative position is important and should be integrated in the feature design.
- In the process of designing a suitable scenario for the multi-agent experiment, it was found that the unpredictable dynamics make it rather hard to design an experiment in which both aircraft operate as is expected when the scenario is designed. If uncertainty arises in the main part of the thesis about inclusion of a new feature in the general multi-agent case experiment, it is advised to construct a scenario in the single-agent setting in which you test the behaviour of the newly synthesized feature.

3.5. Experiment Proposal Final Thesis

The proposal for the subsequent part of the thesis is elaborated upon in this section. The formulation of the experiment design is explained in subsection 3.5.1. Thereafter, the proposed train architecture is discussed in subsection 3.5.2. Finally, subsection 3.5.3 elaborates on the initial model settings for the generalized multi-aircraft experiment.

3.5.1. Experiment Design Thesis

The first part of the proposal consists of the design of experiment structure. The structure should be designed while obeying the following requirements:

- **Traffic Density**: The number and placement of aircraft should facilitate conflicts that involve multiple aircraft.
- **Generalization**: A simulation is not equivalent to reality and assumptions will always be present. However, the layout should strive to mimic a real-life situation and encapture the variability which occurs.
- **Stability**: More variability in the scenarios represents closer resemblance to the true physical encounters which could potentially occur. Nonetheless, variability comes at a price. More variability reduces the stability of the learning process for the agents and requires more exploration. Hence, the experiment structure should make a trade-off between its generalization and stability properties.

The proposed layout of the experiment structure can be found in Figure 3.19 and is based on the experiments performed in [96]. The structure is represented by a hexagon. In each corner an aircraft can be spawned. The target corner of the aircraft can be any corner on the hexagon except for its neighbouring corners. The structure ensures that the aircraft are evenly spawned across the corners of the hexagon and ensures that the aircraft fly towards each other fulfilling the first requirement of traffic density. Albeit aircraft cannot encounter each other from every random angle, the hexagon still allows for a variety in conflict angles, but with a reduced variability. Besides, when aircraft are performing avoidance maneuvers they will deviate from their original heading which results improves the generalization property of the structure.

As explained in chapter 1, the ATM system for UAVs is under development. If the UTM will incorporate fixed air routes with fixed intersections, the experimental structure will accurately mimic the real-life situation. Moreover, delivery drones in cities such as New York city will be forced to follow the roads for cars due to the high skyscrapers which impede the free-flight concept. If the UTM implements a free flight concept, the structure still allows for sufficient variability as explained above.

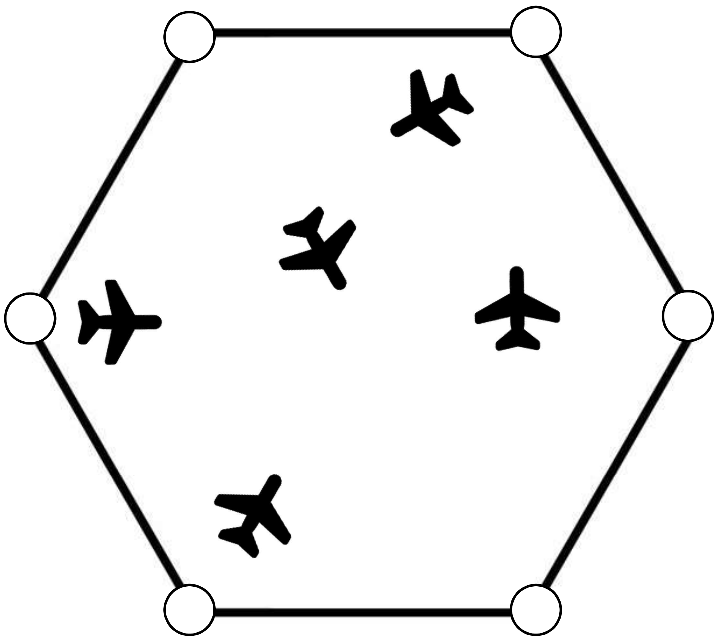


Figure 3.19: Structure of layout from which scenarios are sampled, adapted from [102]

In a scenario, a fixed number of aircraft is spawned and each aircraft is assigned to a different starting node. The target node is randomly selected from the available target nodes. Furthermore, a delay is assigned to each aircraft which prevents the aircraft from all spawning at the same time. The scenario is run untill either all agents have reached their target node or a terminal state is encountered.

3.5.2. Training Architecture

The control scheme of the RL model for the multi-agent experiment was centralized. The centralized controller was implemented to support cooperative behaviour between the agents. In the thesis, a centralized controller is not desired, because it suffers from the curse of dimensionality with an increasing number of agents and it requires communication with a central entity.

The proposed control scheme is centralized learning with decentralized execution and a shared policy $\pi_\theta(s)$. An extensive motivation on the proposed control scheme is provided in subsection 2.3.6. The methodology to update the weights shows great similarity to the approach of algorithm 1. The discrepancy lies in the computation of the advantage function estimates. The advantage function estimates are based on a batch of trajectories from all agents. The centralized learning implies that the agents have a shared reward function and rewards from all trajectories are thus required to construct the advantage estimates. Afterwards, the methodology to update the model parameters is analogous to algorithm 1.

Multi-agent reinforcement learning is a computationally intensive process. Maximizing the computational capacity of the hardware utilized to train the model is beneficial since it reduces the run time. Ray is a distributed framework which empowers systems to cope with intensive parallel simulations of scenarios [59]. The BlueSky architecture has a server-client interface in which several clients can run parallel experiments on separate CPU nodes. Parallel gathering of experience is thus already feasible with the current architecture. The problem is that all agents in the parallel experiments sample from the same policy. Hence, all agents should have access to the policy and the parameter updates of the policy have to be coordinated.

RLlib is part of the Ray project and is a reinforcement learning library which supports high scalability and is able to solve the aforementioned problems concering the central policy. Similar to BlueSky, an option in Ray is to adopt the server-client architecture for the policy. The policy server can be initialized with the policyServerClient class which runs on the local network. Agents in the BlueSky environment can initialize a client with the PolicyClient class. An overview of the architecture is provided by Figure 3.20. The policyServerInput object collects rollout fragments (trajectories) from the environment. Once the pre-specified batch size is attained, the experience is inputted to the trainer object which updates the model parameters. The agents in the BlueSky environment use the PolicyClient to access the current version of the policy. Policy inference can be run with a cached copy of the policy which increases the efficiency of collecting the experience [59].

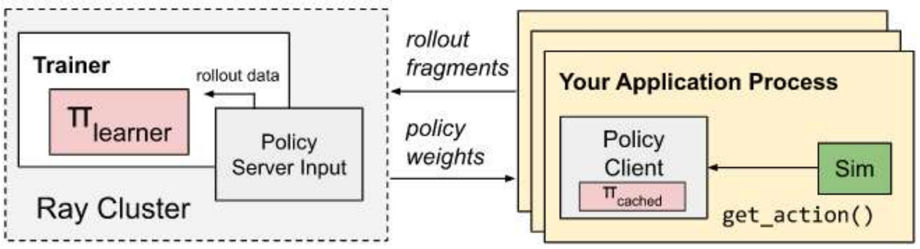


Figure 3.20: Schematic overview of communication between Ray Cluster and environment from [59]

In Figure 3.21, the interaction between the policy server and the BlueSky environment is visualized. Rewards are included in the trajectory data and are computed within the simulation nodes block.

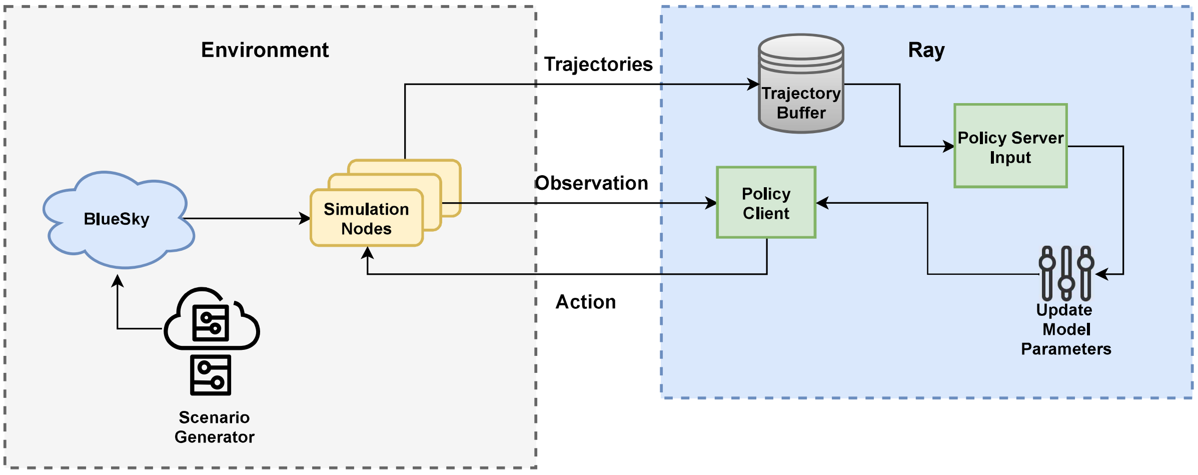


Figure 3.21: Schematic Overview of communication between policy server and BlueSky simulator

3.5.3. Proposed Initial System Settings

Observation Space
In the preliminary thesis, it was decided to discretize the observation space with the idea that it would improve the learning abilities of the PPO algorithm, because the number of possible states is limited. With a continuous state space, the probability that an agent visits a state which it has never encountered is larger, because the number of states is infinite. Nonetheless, the PPO algorithm is able to cope with continuous observation spaces. In the thesis, the approach will be to first implement a discrete action space. Subsequently, a continuous observation space can be implemented to compare performance.

The selected features should be chosen such that no ambiguity exists for the agent. Features should be invariant against rotation and translation. It is hypothesized that the features have to include information on the position of the other aircraft and provide on indicator on how imminent a LoS is. Including the position of other aircraft in a relative positional framework has the advantage that it is automatically invariant against rotation and translation and reduces the necessity to incorporate the state of ownship in the observation vector. This slightly reduces the length of the observation vector which is computationally advantageous.

Utilizing heading to denote the relative position of an intruding aircraft sounds tempting, but is very counterintuitive for the model to interpret. The difference between a heading angle of -179° and 179° is in reality only 2 degrees, but to the model sees it as a difference of of 358°. Therefore, it is advised to implement the relative position feature by decomposing the relative distance into x- and y-components with the relative bearing angle. It is expected that the information on how imminent a conflict is, can be best included with either t_{los} or t_{cpa}.

Due to computational constraints, an increase in the amount of features per aircraft in the observation vector reduces the number of aircraft which can be included in the observation vector. This has to be taken considered when deciding on the number of features per aircraft.

Action Space
The action space will be similar to section 3.3. The action space is not similar to the action space of the multi-agent setting, because a decentralized control scheme is used. The size of the action vector will initially be fixed, because the number of aircraft in a scenario will also be fixed. In later experimental stages in which the number of aircraft is increased, a procedure has to be designed to select the N aircraft which are included in the state space and for which the policy decides on inclusion of the VOs in the SSD.

Reward Function
The various parts of a reward function are determined by the optimization goals which are defined. The goal of this research is to apply conflict prioritization to improve current CD&R methods which incorporate

VOs. It was decided to apply conflict prioritization on the SSD algorithm. The inherent goal of the SSD is to provide a set of ARV which do not result in a LoS. Preventing LoS is therefore an obvious part of the reward function. LoS is a sparse reward and the agents must also aim to optimize its avoidance maneuvers in terms of efficiency. Efficiency will be included in the reward function measured in travelled distance D. No further objectives are initially added to the reward function, because the safety and efficiency objective mimic the goals of the SSD algorithm. Besides, adding more terms to the objective function increases the complexity for the agent which reduces stability. The initial reward function will thus be similar to Equation 3.5. Even though only two objectives are included in the reward function, all performance metrics from Table 2.1.8 can still be utilized to evaluate the method. When the experiments require further design iterations of the reward function, inspiration can be found in [96, 72, 6, 20, 92].

Hyperparameters

In the experiments of the single-agent and multi-agent setting the default hyperparameters were sufficient. Hence, the initial hyperparameter settings for the generalized experiment should also be the default values. A further elaboration on the hyperparameters is provided in section 3.3 and can be examined when tuning of hyperparameters is required.

4
Conclusion

To facilitate an increased number of flights in manned aviation, the concept of Free Flight is proposed which requires a decentralized approach to CD&R. Simultaneously, an increase in the number of drones projected and their obligation to incoporate a DAA system further emphasizes the relevance into decentralized CD&R methods. A class of CD&R algorithms is based on velocity obstacles (VO). When a VO-based method uses a joint solution, the solution space promptly decreases in size with in increase in traffic density. To alleviate this problem, this research proposed to apply conflict prioritization to a VO-based CD&R method.

To proof the veracity of the proposed method, two experiments were conducted based on the theoretical concepts of CD&R and reinforcement learning which were explained in section 2.1, section 2.2 and subsection 2.3.6. It was decided to select the SSD algorithm as the CD&R method to apply conflict prioritization on.

The preliminary experiments were conducted in a single-agent setting and a multi-agent setting in chapter 3. The RL model controls one aircraft in the single-agent setting and two in the multi-agent setting. The PPO algorithm was concluded to be to most suitable for this research and thus selected as RL model. In the multi-agent experiment the PPO model was implemented with a centralized control scheme. In both settings, the agents were successfully trained to prevent LoS while increasing the efficiency of the resolution maneuver.

In the subsequent part of the thesis, the focus will be to extend the proof-of-concept from the single- and multi-agent setting to a generalized multi-agent experiment with a higher number of controlled aircraft. The variability in the scenarios is increased which allows the agents to encounter more conflict scenarios with a different conflict geometry. In the initial setup the PPO method will be implemented with a decentralized control scheme and a shared policy to deal with the challenges of multi-agent reinforcement learning. However, issues are still expected to arise from the increased complexity of interacting agents which are evolving simultaneously. An important aspect will be to fine-tune the model settings by iteratively updating of the model settings and evaluation of the dependent variables.

Bibliography

[1] “Right-of-Way Rules: Except Water Operations”. *Air Traffic and General Operating Rules.* Federal Aviation Administration Regulation, Title 14, Chap. 1.F, Pt. 91.B, Sec. 91.113, July 2004.

[2] S. Balasooriyan. “Multi-aircraft conflict Resolution using Velocity Obstacles”. Master’s Thesis. Delft University of Technology, 2017.

[3] M. G. Bellemare et al. “The Arcade Learning Environment: An Evaluation Platform for General Agents”. In: *Journal of Artificial Intelligence Research* 47 (June 2013), pp. 253–279. DOI: 10.1613/jair.3912.

[4] J. Bergstra and Y. Bengio. “Random search for hyper-parameter optimization.” In: *Journal of machine learning research* 13.2 (2012).

[5] C. Borst, C. Westin, and B. Hilburn. “An investigation into the use of novel conflict detection and resolution automation in air traffic management”. In: *SIDs 2012 - Proceedings of the SESAR Innovation Days* November (2012).

[6] M. Brittain, X. Yang, and P. Wei. *A Deep Multi-Agent Reinforcement Learning Approach to Autonomous Separation Assurance.* 2020. arXiv: 2003.08353.

[7] G. Brockman et al. *OpenAI Gym.* 2016. arXiv: 1606.01540.

[8] L. Buşoniu, R. Babuška, and B. De Schutter. “A comprehensive survey of multiagent reinforcement learning”. In: *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* 38.2 (2008), pp. 156–172. ISSN: 10946977. DOI: 10.1109/TSMCC.2007.913919.

[9] L. Buşoniu, R. Babuška, and B. De Schutter. “Multi-agent reinforcement learning: An overview”. In: *Innovations in multi-agent systems and applications-1* (2010), pp. 183–221.

[10] S. Cafieri. “Chapter 22: Mixed-Integer Nonlinear Optimization in Air Traffic Management: Aircraft Conflict Avoidance”. In: *Advances and Trends in Optimization with Engineering Applications.* Society for Industrial and Applied Mathematics, Apr. 2017, pp. 293–301. DOI: 10.1137/1.9781611974683.ch22.

[11] CBS. *Aviation: monthly figures of Dutch airports.* URL: https://opendata.cbs.nl/statline/#/CBS/en/dataset/37478eng/table. 2021.

[12] P. Conroy et al. *3-D Reciprocal Collision Avoidance on Physical Quadrotor Helicopters with On-Board Sensing for Relative Positioning.* 2014. arXiv: 1411.3794.

[13] M. Doole, J. Ellerbroek, and J. Hoekstra. “Estimation of traffic density from drone-based delivery in very low level urban airspace”. In: *Journal of Air Transport Management* 88.June (2020), p. 101862. ISSN: 09696997. DOI: 10.1016/j.jairtraman.2020.101862.

[14] M. Doole, J. Ellerbroek, and J. Hoekstra. “Drone delivery: Urban airspace traffic density estimation”. In: *8th SESAR Innovation Days, 2018* (2018).

[15] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright. “Randomized smoothing for (parallel) stochastic optimization”. In: *Proceedings of the IEEE Conference on Decision and Control* 12 (2012), pp. 5442–5444. ISSN: 01912216. DOI: 10.1109/CDC.2012.6426698.

[16] J. Ellerbroek. “Airborne Conflict Resolution In Three Dimensions””. PhD thesis. Delft University of Technology, 2013.

[17] H. Emami, F. Derakhshan, and S. Pashazadeh. “A new prioritization method for conflict detection and resolution in air traffic management”. In: *Journal of Emerging Trends in Computing and Information Sciences* 3.7 (2012), pp. 1042–1049.

[18] EUROCONTROL. *Model for Task and Job Descriptions of Air Traffic Controllers.* European AirTraffic Control Harmonisation and Integration Programme, 1996.

[19] EUROCONTROL. *Performance Review Report An Assessment of Air Traffic Management in Europe during the Calendar Year 2018.* 2018.

[20] T. Fan et al. *Fully Distributed Multi-Robot Collision Avoidance via Deep Reinforcement Learning for Safe and Efficient Navigation in Complex Scenarios.* 2018. eprint: `arXiv:1808.03841`.

[21] P. Fiorini and Z. Shiller. "Motion Planning in Dynamic Environments Using Velocity Obstacles". In: *The International Journal of Robotics Research* 17.7 (1998), pp. 760–772. DOI: `10.36288/roscon2012-900669`.

[22] J. Foerster et al. "Stabilising experience replay for deep multi-agent reinforcement learning". In: *34th International Conference on Machine Learning, ICML 2017* 3 (2017), pp. 1879–1888. arXiv: `1702.08887`.

[23] J. K. Gupta, M. Egorov, and M. Kochenderfer. "Cooperative Multi-agent Control Using Deep Reinforcement Learning". In: *Autonomous Agents and Multiagent Systems* (2017), pp. 66–83. DOI: `10.1007/978-3-319-71682-4`.

[24] M. Haenlein and A. Kaplan. "A brief history of artificial intelligence: On the past, present, and future of artificial intelligence". In: *California Management Review* 61.4 (2019), pp. 5–14. ISSN: 21628564. DOI: `10.1177/0008125619864925`.

[25] S. HAO, S. CHENG, and Y. ZHANG. "A multi-aircraft conflict detection and resolution method for 4-dimensional trajectory-based operation". In: *Chinese Journal of Aeronautics* 31.7 (July 2018), pp. 1579–1593. DOI: `10.1016/j.cja.2018.04.017`.

[26] P. Hermes et al. "Solution-space-based analysis of the difficulty of aircraft merging tasks". In: *Journal of Aircraft* 46.6 (2009), pp. 1995–2015. ISSN: 15333868. DOI: `10.2514/1.42886`.

[27] B. Hilburn. "Cognitive complexity in air traffic control: A literature review". In: *EEC note* 4.04 (2004), pp. 1–80.

[28] A. Hill et al. *Stable Baselines.* `https://github.com/hill-a/stable-baselines`. 2018.

[29] F. Ho et al. "Decentralized Multi-Agent Path Finding for UAV Traffic Management". In: *IEEE Transactions on Intelligent Transportation Systems* (2020), pp. 1–12. DOI: `10.1109/TITS.2020.3019397`.

[30] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. ISSN: 08997667. DOI: `10.1162/neco.1997.9.8.1735`.

[31] J. M. Hoekstra, R. N. Van Gent, and R. C. Ruigrok. "Designing for safety: The 'free flight' air traffic management concept". In: *Reliability Engineering and System Safety* 75.2 (2002), pp. 215–232. ISSN: 09518320. DOI: `10.1016/S0951-8320(01)00096-5`.

[32] J. Hoekstra, R. Ruigrok, and R. Van Gent. "Free flight in a crowded airspace?" In: *Proceedings of the 3rd USA/Europe Air Traffic Management R&D Seminar.* 2001.

[33] P. J. '. Hoen et al. "An Overview of Cooperative and Competitive Multiagent Learning BT - Learning and Adaption in Multi-Agent Systems". In: (2006), pp. 1–46.

[34] C. C.-Y. Hsu, C. Mendler-Dünner, and M. Hardt. *Revisiting Design Choices in Proximal Policy Optimization.* 2020. arXiv: `2009.10897`.

[35] J. Hu, M. Prandini, and S. Sastry. "Optimal coordinated maneuvers for three-dimensional aircraft conflict resolution". In: *Journal of Guidance, Control, and Dynamics* 25.5 (2002), pp. 888–900.

[36] Y. Huang. "Deep Q-networks". In: *Deep Reinforcement Learning: Fundamentals, Research and Applications* (2013), pp. 135–160. DOI: `10.1007/978-981-15-4095-0_4`.

[37] Y. Huo, D. Delahaye, and Y. Wang. "Sensitivity Analysis of Closest Point of Approach". In: *8th International Conference for Research in Air Transportation,* (June 2018).

[38] ICAO. *Doc 4444 - PANS-ATM, Procedures for Navigation Services – Air Traffic Management.* 16th ed. International Civil Aviation Organisation, 2016.

[39] ICAO. *Doc 9882 AN/467, Manual on Air Traffic Management System Requirements.* 1st ed. 2008.

[40] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *32nd International Conference on Machine Learning, ICML 2015* 1 (2015), pp. 448–456. arXiv: `1502.03167`.

[41] J. E. Jacco Hoekstra. "BlueSky ATC Simulator Project: An Open Data and Open Source Approach". In: *7th International Conference on Research in Air Transportation: Philadelphia, USA* (2016).

[42] Joint Planning and Development Office (JPDO) and Next Generation Air Transportation System (NextGen). *Concept of operations for the next generation air transportation system.* Tech. rep. 2011.

[43] A. Kaplan and M. Haenlein. "Siri, Siri, in my hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence". In: *Business Horizons* 62.1 (Jan. 2019), pp. 15–25. DOI: 10.1016/j.bushor.2018.08.004.

[44] D. P. Kingma and J. L. Ba. "Adam: A method for stochastic optimization". In: *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* (2015), pp. 1–15. arXiv: 1412.6980.

[45] J. Klooster et al. "Trajectory synchronization and negotiation in trajectory based operations". In: *29th Digital Avionics Systems Conference.* IEEE. 2010, 1.A.3-1–1.A.3–11.

[46] A. Koul, S. Greydanus, and A. Fern. *Learning Finite State Representations of Recurrent Policy Networks.* 2018. arXiv: 1811.12530.

[47] J. K. Kuchar and L. C. Yang. "A Review of Conflict Detection and Resolution Modeling Methods". In: *IEEE Transactions on Intelligent Transportation Systems* 1.4 (2000), pp. 179–189. ISSN: 15249050. DOI: 10.1109/6979.898217.

[48] J. Kuchar and A. C. Drumm. "The traffic alert and collision avoidance system". In: *Lincoln laboratory journal* 16.2 (2007), p. 277.

[49] T. Langejan. "Effect of ADS-B Limitations and Inaccuracies on CDR Performance". 2016.

[50] Y. Lecun, Y. Bengio, and G. Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444. ISSN: 14764687. DOI: 10.1038/nature14539.

[51] T. P. Lillicrap et al. "Continuous control with deep reinforcement learning". In: *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings* (2016). arXiv: 1509.02971.

[52] L.-J. Lin. "Self-improving reactive agents based on reinforcement learning, planning and teaching". In: *Machine Learning* 8.3-4 (May 1992), pp. 293–321. DOI: 10.1007/bf00992699.

[53] M. L. Littman. *Markov games as a framework for multi-agent reinforcement learning.* Morgan Kaufmann Publishers, Inc., 1994, pp. 157–163. DOI: 10.1016/b978-1-55860-335-6.50027-1. URL: http://dx.doi.org/10.1016/B978-1-55860-335-6.50027-1.

[54] R. Lowe et al. "Multi-agent actor-critic for mixed cooperative-competitive environments". In: *Advances in Neural Information Processing Systems* 2017-Decem (2017), pp. 6380–6391. ISSN: 10495258. arXiv: 1706.02275.

[55] X. Ma et al. "3-D Decentralized Prioritized Motion Planning and Coordination for High-Density Operations of Micro Aerial Vehicles". In: *IEEE Transactions on Control Systems Technology* 26.3 (2018), pp. 939–953. DOI: 10.1109/TCST.2017.2699165.

[56] G. A. Mercado Velasco, M. Mulder, and M. M. Van Paassen. "Analysis of air traffic controller workload reduction based on the solution space for the merging task". In: *AIAA Guidance, Navigation, and Control Conference* August (2010). DOI: 10.2514/6.2010-7541.

[57] V. Mnih et al. "Asynchronous methods for deep reinforcement learning". In: *33rd International Conference on Machine Learning, ICML 2016* 4 (2016), pp. 2850–2869. arXiv: 1602.01783.

[58] T. Molloy et al. "Evaluation of the Rules of the Air for the Future of Air Traffic Management". In: *AIAA AVIATION 2020 FORUM.* American Institute of Aeronautics and Astronautics, June 2020. DOI: 10.2514/6.2020-2855.

[59] P. Moritz et al. "Ray: A distributed framework for emerging AI applications". In: *Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018* (2007), pp. 561–577. arXiv: 1712.05889.

[60] NextGEN. *Concept of Operations v2.0.* https://www.faa.gov/uas/research_development/traffic_management/media/UTM_ConOps_v2.pdf. 2020.

[61] H. Niu, C. Ma, and P. Han. "A decentralized method for collision detection and avoidance applied to civil aircraft". In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* (2020). ISSN: 20413025. DOI: 10.1177/0954410020953045.

[62] H. Niu, C. Ma, and P. Han. “A decentralized method for collision detection and avoidance applied to civil aircraft”. In: *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* (2020). ISSN: 20413025. DOI: 10.1177/0954410020953045.

[63] C. Olah. *Understanding LSTM Networks*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. 2015.

[64] A. OroojlooyJadid and D. Hajinezhad. “A review of cooperative multi-agent deep reinforcement learning”. In: *arXiv* (2019). ISSN: 23318422. arXiv: 1908.03963.

[65] L. Pallottino, E. M. Feron, and A. Bicchi. “Conflict resolution problems for air traffic management systems solved with mixed integer programming”. In: *IEEE transactions on intelligent transportation systems* 3.1 (2002), pp. 3–11.

[66] G. Papoudakis et al. *Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks*. 2020. arXiv: 2006.07869.

[67] A. Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[68] L. P. I. da Piedade. “Aircraft Conflict Prioritization and Resolution using the Solution Space Diagram”. 2018.

[69] *PRONETOWANDERFRIENDS’S BLOG*. URL: https://pronetowanderfriends.wordpress.com/2014/02/23/angels-landing-in-zion-national-park-check/. 2014.

[70] S. A. Rahman et al. “Cross-sector transferability of metrics for air traffic controller workload”. In: *IFAC-PapersOnLine* 49.19 (2016), pp. 313–318. DOI: 10.1016/j.ifacol.2016.10.561. URL: https://doi.org/10.1016/j.ifacol.2016.10.561.

[71] M. Ribeiro, J. Ellerbroek, and J. Hoekstra. “Determining Optimal Conflict Avoidance Manoeuvres At High Densities With Reinforcement Learning”. In: December (2020).

[72] M. Ribeiro, J. Ellerbroek, and J. Hoekstra. “Improvement of Conflict Detection and Resolution at High Densities Through Reinforcement Learning”. In: *9th International Conference for Research in Air Transportation (ICRAT)* (2020).

[73] M. Ribeiro, J. Ellerbroek, and J. Hoekstra. “Review of conflict resolution methods for manned and unmanned aviation”. In: *Aerospace* 7.6 (2020). ISSN: 22264310. DOI: 10.3390/AEROSPACE7060079.

[74] H. Robbins and S. Monro. “A Stochastic Approximation Method”. In: *The Annals of Mathematical Statistics* 22.3 (1951), pp. 400–407. DOI: 10.1214/aoms/1177729586. URL: https://doi.org/10.1214/aoms/1177729586.

[75] S. Ruder. “An overview of gradient descent optimization algorithms”. In: (2016), pp. 1–14. arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[76] J. Schulman et al. “High-dimensional continuous control using generalized advantage estimation”. In: *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings* (2016), pp. 1–14. arXiv: 1506.02438.

[77] J. Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347.

[78] J. Schulman et al. “Trust region policy optimization”. In: *32nd International Conference on Machine Learning, ICML 2015* 3 (2015), pp. 1889–1897. arXiv: 1502.05477.

[79] SESAR Consortium. *The Concept of Operations at a Glance*. Single European Sky. 2007.

[80] SESAR Joint Undertaking. *U-space: blueprint*. Publications Office, 2017. DOI: 10.2829/335092.

[81] SESAR joint Undertaking. *Bubbles Separation Management*. https://www.sesarju.eu/projects/bubbles. 2020.

[82] SESAR joint Undertaking. *European Drones Outlook Study Unlocking the value for Europe*. https://www.sesarju.eu/sites/default/files/documents/reports/European_Drones_Outlook_Study_2016.pdf. 2016.

[83] D. Silver. *Lectures on Reinforcement Learning*. URL: https://www.davidsilver.uk/teaching/. 2015.

[84] D. Silver et al. “Deterministic policy gradient algorithms”. In: *31st International Conference on Machine Learning, ICML 2014* 1 (2014), pp. 605–619.

[85] D. Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), pp. 484–489.

[86] J. Sun. “Open Aircraft Performance Modeling: Based on an Analysis of Aircraft Surveillance Data”. PhD thesis. 2019. DOI: 10.4233/UUID:AF94D535-1853-4A6C-8B3F-77C98A52346A.

[87] E. Sunil et al. “Analysis of airspace structure and capacity for decentralized separation using fast-time simulations”. In: *Journal of Guidance, Control, and Dynamics* 40.1 (2017), pp. 38–51.

[88] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 2018.

[89] M. Tan. “Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents”. In: *Machine Learning Proceedings 1993* (1993), pp. 330–337. DOI: 10.1016/b978-1-55860-307-3.50049-6.

[90] T. Tieleman and G. Hinton. *Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning.* URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. 2012.

[91] M. Tra. “The Effect of a Layered Airspace Concept on Conflict Probability and Capacity”. MA thesis. 2016.

[92] D. Q. Tran and S. H. Bae. “Proximal policy optimization through a deep reinforcement learning framework formultiple autonomous vehicles at a non-signalized intersection”. In: *Applied Sciences (Switzerland)* 10.16 (2020). ISSN: 20763417. DOI: 10.3390/app10165722.

[93] S. B. Van Dam et al. “Functional presentation of travel opportunities in flexible use airspace: An EID of an airborne conflict support tool”. In: *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics* 1 (2004), pp. 802–808. ISSN: 1062922X. DOI: 10.1109/ICSMC.2004.1398401.

[94] J. Van Den Berg et al. “Reciprocal n-body collision avoidance”. In: *Robotics research.* Springer, 2011, pp. 3–19.

[95] O. Vinyals et al. *StarCraft II: A New Challenge for Reinforcement Learning.* 2017. arXiv: 1708.04782.

[96] D. Wang et al. “A Two-Stage Reinforcement Learning Approach for Multi-UAV Collision Avoidance under Imperfect Sensing”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 3098–3105. ISSN: 23773766. DOI: 10.1109/LRA.2020.2974648.

[97] C. Watkinsk. “Learning from Delayed Rewards”. PhD thesis. University of Cambridge, 1989.

[98] J. Wu et al. “Hyperparameter optimization for machine learning models based on Bayesian optimization”. In: *Journal of Electronic Science and Technology* 17.1 (2019), pp. 26–40.

[99] Y. Wu et al. “Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation”. In: *Advances in Neural Information Processing Systems* 2017-December (2017), pp. 5280–5289. ISSN: 10495258. arXiv: 1708.05144.

[100] J. Yang et al. “Distributed cooperative onboard planning for the conflict resolution of unmanned aerial vehicles”. In: *Journal of Guidance, Control, and Dynamics* 42.2 (2019), pp. 272–283. ISSN: 15333884. DOI: 10.2514/1.G003583.

[101] H. Zhang and T. Yu. “Taxonomy of Reinforcement Learning Algorithms”. In: *Deep Reinforcement Learning.* Springer Singapore, 2020, pp. 125–133. DOI: 10.1007/978-981-15-4095-0_3.

[102] Y. Zhao et al. “Reinforcement Learning-Based Collision Avoidance Guidance Algorithm for Fixed-Wing UAVs”. In: *Complexity* (2021). ISSN: 10990526. DOI: 10.1155/2021/8818013.