Aerospace Technology Congress, 8-9 October 2019, Stockholm, Sweden
Swedish Society of Aeronautics and Astronautics (FTF)

Multi-Agent Multi-Objective Deep Reinforcement Learning for Efficient and Effective Pilot Training

Johan Källström[*] and Fredrik Heintz[**]

[*]Saab AB and Department of Computer Science, Linköping University, Linköping, Sweden
[**]Department of Computer Science, Linköping University, Linköping, Sweden
E-mail: Johan.Kallstrom@liu.se, Fredrik.Heintz@liu.se

Abstract

The tactical systems and operational environment of modern fighter aircraft are becoming increasingly complex. Creating a realistic and relevant environment for pilot training using only live aircraft is difficult, impractical and highly expensive. The Live, Virtual and Constructive (LVC) simulation paradigm aims to address this challenge. LVC simulation means linking real aircraft, ground-based systems and soldiers (Live), manned simulators (Virtual) and computer controlled synthetic entities (Constructive). Constructive simulation enables realization of complex scenarios with a large number of autonomous friendly, hostile and neutral entities, which interact with each other as well as manned simulators and real systems. This reduces the need for personnel to act as role-players through operation of e.g. live or virtual aircraft, thus lowering the cost of training. Constructive simulation also makes it possible to improve the availability of training by embedding simulation capabilities in live aircraft, making it possible to train anywhere, anytime. In this paper we discuss how machine learning techniques can be used to automate the process of constructing advanced, adaptive behavior models for constructive simulations, to improve the autonomy of future training systems. We conduct a number of initial experiments, and show that reinforcement learning, in particular multi-agent and multi-objective deep reinforcement learning, allows synthetic pilots to learn to cooperate and prioritize among conflicting objectives in air combat scenarios. Though the results are promising, we conclude that further algorithm development is necessary to fully master the complex domain of air combat simulation.

Keywords: Pilot Training, Embedded Training, LVC Simulation, Artificial Intelligence, Autonomy, Sub-system and System Technology, Aircraft and Spacecraft System Analysis

1 Introduction

The tactical systems and operational environment of modern fighter aircraft are becoming increasingly complex. As a consequence, conducting training using only live, manned platforms is becoming increasingly difficult. Live training is related to high costs, and air space regulations as well safety restrictions place limitations on the type of training scenarios that can be realized. The logistics related to live training may also lead to poor availability of training. As the possibilities to do live training decrease, simulation-based training becomes more and more important.

In an ongoing project within the Swedish National Aeronautical Research Program 7 (NFFP7), we are studying how the next-generation pilot training systems should be designed to meet future training needs. In our research we are investigating how machine learning techniques can be used to construct advanced behavior models for synthetic, intelligent agents. The goal is to develop efficient methods to generate a wide range of intelligent, adaptive computer controlled allies and adversaries that can create realistic situations adapted for training of fighter pilots. We have identified two subfields of particular interest: Multi-Agent Reinforcement Learning (MARL) and Multi-Objective Reinforcement Learning (MORL) [1, 2]. MARL allows agents to learn how to achieve their goals in mixed cooperative and competitive multi-agent scenarios, such as an air combat scenario, while MORL allows agents to learn how to prioritize among multiple conflicting objectives, e.g. tactical mission goals, resource consumption and safety.

In this paper we discuss how these techniques can help address the challenges related to constructing high quality training simulations. We first give an overview of simulation-based training, and highlight aspects that motivate our work. We then present a proposed architecture for an intelligent, synthetic trainer, and machine learning techniques that could be used to implement it. Finally, we evaluate the approach in a number of experiments, with promising results, and give directions for future work.

2 Simulation-Based Training

Simulation-based training is an efficient way of training operators in complex, high-risk tasks. Scenarios with great variety can be realized at a low cost, without risk of injury. In our work, we model a typical training process for a simulation-based training system as illustrated in fig. 1. First, simulation contents are created to meet identified training needs. In the domain of air combat this could include vehicle models, behavior models for the synthetic operators of these vehicles, and definitions of the scenarios that they operate in. Then, in a training session, a briefing is conducted to present and discuss training objectives and scenario contents, followed by the actual execution of the scenario. Afterwards, trainee performance is evaluated in a debriefing. Over time, training needs are updated based on the learning progress of trainees, as well as input from the organization that they belong, e.g. due to changes in operational missions.

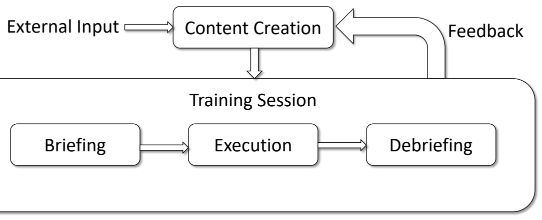

Figure 1: Training process.

User roles associated with the training process are illustrated in fig. 2. The Training Audience are those that we want to train, in the case of air combat training fighter pilots. The training environment is provided by a set of Training Providers. The Instructor is responsible for the pedagogical contents of a training session, and is supported by role-players and operators to provide it. Role-players participate in the training scenario, but they themselves do not receive training. Operators work behind the scenes of the training scenario, e.g. controlling simulation software, such as manual control of Computer Generated Forces (CGF), to make sure that the simulated scenario progresses in the right direction. In practice, one single person could act in several roles. For instance, due to limited resources, one person could act as instructor role-player and operator. This typically results in a high workload, and the desired training scenarios may not be achievable. It is desirable to reduce the need for training providers, to improve training efficiency as well as effectiveness.

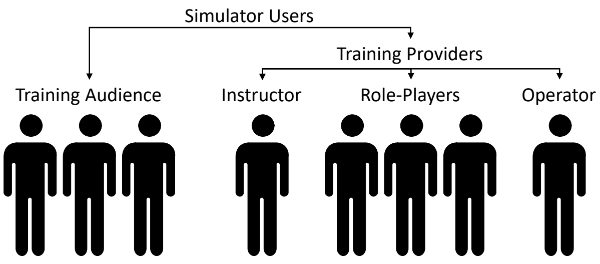

Figure 2: Users of simulation-based training systems.

2.1 Live, Virtual and Constructive Simulation

Computer simulations can be used to augment training in live systems. One approach is to replace some of the live training with training in simulators, e.g. using ground-based flight simulators instead of training in live aircraft. Another approach is to embed simulation capabilities in live systems, e.g. capabilities for generation of synthetic opponents in a fighter aircraft. It has been estimated that embedded training can improve training effectiveness of live training by 30% at the same cost [3]. In the Live, Virtual and Constructive simulation paradigm, the goal is to take things one step further, by seamlessly integrating live systems, manned simulators and computerized simulations in a distributed simulation. The three categories of simulations are defined as [4]:

- *Live*: Simulations involving real people operating real systems
- *Virtual*: Simulations involving real people operating simulated systems
- *Constructive*: Simulations involving simulated people operating simulated systems (possibly stimulated by real people)

In the domain of air combat simulation the goal is to integrate real aircraft, ground-based systems and soldiers (Live), manned simulators (Virtual) and computer controlled entities (Constructive). Such a simulation platform is valuable for training [5,6]. An example of an LVC simulation network for air combat training is illustrated in fig. 3.

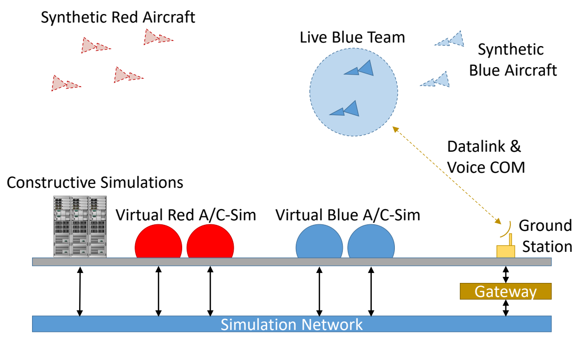

Figure 3: An LVC distributed simulation.

Constructive simulation enables realization of complex scenarios with a large number of autonomous friendly, hostile and neutral entities, which interact with each other as well as manned simulators and real systems. However, building realistic behavior models for CGF is a significant challenge [7–9], and consequently support from scenario operators, and possibly human role-players, is still required in many training scenarios. With improved behavior models, training systems with a higher level of autonomy could be built, and adaptive training (AT) with contents tailored to the current learning needs of individual trainees could be provided [10].

3 An Intelligent, Synthetic Trainer

To make pilot training more efficient and effective, we would like to increase the autonomy of air combat training systems, and minimize the dependence on human training providers. For this purpose, we propose to construct an intelligent, synthetic trainer, which can learn to understand the learning needs of trainees, and then act accordingly to provide the best possible training, as well as support for evaluation of trainees. The synthetic trainer should be able to represent allies as well as adversaries in a training scenario. We are investigating how machine learning could be used to create such an agent. A proposed architecture is shown in fig. 4.

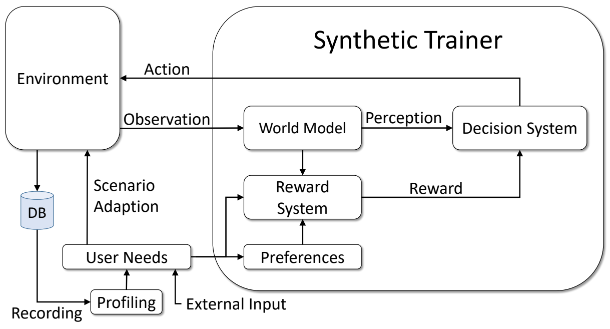

Figure 4: Architecture of a Synthetic Trainer.

The agent interacts with an environment, populated by human trainees as well as other, synthetic agents, who participate in an air combat training scenario. Users' training needs must be considered by a synthetic agent when acting in this scenario, and thus affect the goals of the agent, as well as preferences regarding how the agent should try to achieve those goals. In current training systems user needs are typically handled manually by an instructor, e.g. when constructing simulation contents or conducting training sessions, but we would like to automate this process, by using machine learning to model the training needs, progress and proficiency of the trainee. These models could be constructed between training sessions, based on recordings, or online during the execution of a training sessions, based on the observations of the agent. They can then be used to automatically adapt the contents and characteristics of the training scenario.

The agent's observations of the world are used as input to a world model, which will create the higher level perception of the agent. Such models can be used to predict the winner of a game given a certain state, as well as the skills, beliefs, long term goals and immediate actions of agents [11–17]. Parts of the model could be constructed by hand, based on domain knowledge, but we are primarily investigating learning approaches, such as Supervised Deep Learning or Unsupervised Learning [18, 19]. Since data from real or simulated air battles, with aircraft operated by human pilots, are not readily available, the intention is to use synthetic data for training of machine learning models.

The decision system has capabilities for learning policies suitable for training of trainees. These policies should consider the goals of the agent in the simulated air combat scenario, as well as the learning objectives of the trainees. As mentioned, limited data is available from human pilots. Instead we have identified reinforcement learning as a promising technique for implementing this system, since it allows an agent to learn based on interaction with a simulation. Feedback regarding the agent's learning progress is then provided by a reward system. Reinforcement learning is discussed further in the following sections.

3.1 Reinforcement Learning

Reinforcement learning is a machine learning paradigm, which aims to allow synthetic agents to learn how to achieve their goals by interacting with their environment [20]. In recent years the technique has had great success in training agents to solve games, such as Go and StarCraft [11, 12, 21–23], as well as complex control tasks [24–26].

Reinforcement learning problems are modelled as Markov Decision Processes (MDP). An MDP is a tuple (S, A, T, R, γ), specifying:

- S: The finite set of states of the process
- A: The finite set of actions of the process
- T: The transition dynamics of the process
- R: The reward function of the process
- γ: The discount factor indicating the importance of immediate and future rewards respectively

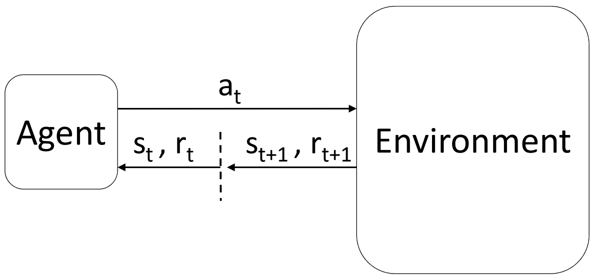

Figure 5: Markov Decision Process.

As illustrated in in fig. 5, in each time step the agent selects an action, observes the resulting new state of the environment, and receives a reward. The objective of the agent is to maximize its future expected return:

$$V_\pi(s) = E[R_t | s_0 = s] = E[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s] \qquad (1)$$

The state value function $V_\pi(s)$ specifies the value of being in state s and then following policy π. Similarly, the value of being in state s and taking action a, and then following policy π is given by the state-action value function:

$$Q_\pi(s, a) = E[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a] \qquad (2)$$

One popular algorithm for reinforcement learning is Q-learning [27], which seeks to estimate the state-action value function *Q(s,a)*. This algorithm was extended to handle complex, continuous state spaces in the Deep Q-Networks algorithm [28], which uses deep neural networks to represent the agent's policy.

3.2 Multi-Agent Reinforcement Learning

In most air combat scenarios, pilots do not act on their own, but instead must cooperate with allies to achieve their goals, while competing with enemies. To train teams of agents, multi-agent reinforcement learning can be used. The single agent MDP can be extended to include multiple agents in so called Stochastic Games (SG), where multiple agents interact with the environment, and the environment state as well as the rewards of individual agents are determined by the joint actions of all agents [1]. Stochastic games can be characterized as fully cooperative when all agents have the same goal, and fully competitive when agents have opposite goals. Stochastic games that are neither fully cooperative nor fully competitive are called mixed games. A special case of cooperative stochastic games is the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [29,30]. A Dec-POMDP is a tuple $(S,\{A_i\},T,R,\{\Omega_i\},O,\gamma)$, specifying:

- S: The finite set of states of the process
- A_i: The finite set of actions of agent i
- T: The transition dynamics of the process
- R: The reward function of the process
- Ω_i: The finite set of observations of agent i
- O: The finite set of conditional observation probabilities
- γ: The discount factor indicating the importance of immediate and future rewards respectively

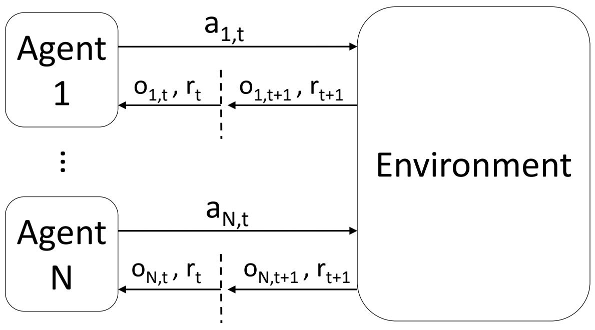

Figure 6: Decentralized Partially Observable Markov Decision Process.

Agents in the Dec-POMD, illustrated in fig. 6, can only observe parts of the environment state, since e.g. internal states of other agents may be hidden. In an air combat scenario observability could be affected by e.g. performance of sensors and data links, which could also be affected by electronic warfare. In the illustrated Dec-POMDP, agents must try to coordinate their actions to maximize a shared reward. In other settings agents could have individual rewards, e.g. due to different priorities among members in a team, or due to a competitive scenario.

Multi-agent learning presents many challenges, such as coordination among agents and multi-agent credit assignment (i.e. determining a single agent's contribution to the success of a team of agents). As several agents learn concurrently, the environment may also become non-stationary from a single agent's point of view. The Multi-Agent Deep Deterministic Policy Gradient algorithm (MADDPG) [31] uses a centralized Q function to guide updates of decentralized policies. The algorithm supports continuous action spaces and mixed cooperative-competitive scenarios, and is thus suitable for applications in the air combat domain.

Multi-agent reinforcement learning can also be used as a framework for creating hierarchical policies for single entities in a simulation [32, 33], by placing agents in a hierarchy where higher-level agents try to reach abstract goals by issuing commands to lower level agents, with increasingly reactive behavior.

3.3 Multi-Objective Reinforcement Learning

Multi-objective reinforcement learning can be used to learn policies for problems where multiple, possible conflicting objectives must be considered [2]. Typical air combat scenarios fit this description, since they require that the participating pilots prioritize among objectives such as targets to attack, assets to protect, safety and resource consumption. In training scenarios, synthetic agents could also consider the learning objective of trainees, e.g. by adapting their behavior to fit the proficiency of the trainee. In multi-objective reinforcement learning the single-objective MDP is extended to a Multi-Objective Markov Decision Process (MOMDP). An MOMDP provides a vector-valued reward function, with each element representing the reward for one of the objectives. The user utililty of the vector-valued return of an MOMDP is given by using a scalarization function, which converts the vector to a scalar. One option is to use the weighted sum of all values:

$$V_{\mathbf{w}}^{\pi}(s) = f(\mathbf{V}^{\pi}(s),\mathbf{w}) = \sum_{n=1}^{n} v_i^{\pi}(s) w_i \tag{3}$$

One approach for solving an MOMDP is to use scalarization directly on the vector-valued reward signal, to convert the MOMDP to an MDP for a set of preferences, and then use single-objective methods to find a set of policies [34, 35]. Then, at execution time, the user can select a suitable policy. By using a stochastic mixture of policies over time for episodic tasks, i.e. selecting one of several policies by random before the start of an episode, further parts of the solution space can be covered [36]. In our previous work, we proposed an approach for training a single, tunable neural network policy to prioritize among a set of objectives at execution time, by conditioning the network on user preferences [37].

4 Experiments

To evaluate the potential of machine learning, in particular multi-agent reinforcement learning and multi-objective reinforcement learning, as a tool for building CGF behavior models, we conduct a number of experiments. As simulation platform we use the tactical environment simulation software that is part of the Saab Gripen Flight Training Simulators. All simulation results are averaged over five runs with different random seeds.

4.1 Coordination of a Tactical Air Unit

We first study how multi-agent reinforcement learning can be used to coordinate the actions of agents that are members of the same Tactical Air Unit (TAU). For this purpose, we use the MADDPG algorithm. We also use environments that are similar to those used in the original paper [31], but implemented in our high-fidelity simulation engine. The increased complexity of the state space, as well as the increased number of time steps per simulated episode, add additional difficulty. We investigate how the algorithm performs in this setting, for different types of action spaces. The policy is represented by a multilayer perceptron (MLP), with 2 hidden layers, each with 64 neurons and the ReLU activation function. We use a learning rate of $\alpha = 10^{-2}$, a discount factor of $\gamma = 0.95$, and train using the Adam optimizer.

4.1.1 Coordinated Defense

In this scenario there are three high-value assets that should be protected by three learning agents. The assets are attacked by three enemy agents, which are controlled by handcrafted behavior models, implemented with Behavior Trees [38]. If a defending agent comes within 5 km of an attacking agent, the attacker will retreat to its home base, and then attack again. To protect all three high-value assets, the learning agents must learn to split up and escort one enemy each from the protected area. The defending agents are initialized with random positions and headings, while the attacking agents are initialized at random positions along their planned attack routes. The spawn area of blue aircraft and attack directions of red aircraft are shown in fig. 7.

The observation space of each agent is the relative position of all other agents, in a body-fixed coordinate system, for the last 4 time steps in the episode. We study three types of action spaces. The first two are continuous action spaces, which allow an agent to fly forward, or turn left or right with a load factor of 2-4 g. One of these action spaces also allows an agent to set its internal state as a three element, real-valued and normalized vector, which is then distributed to the other agents in the team in each time step. Previous work has shown that this type of mechanism can allow agents to develop a language for coordination of their actions [31]. The third action space is a hierarchical approach, with discrete actions that let the agent select a target and assign it as goal for a lower level controller. For the first two types of action spaces, the agent is executed at 1 s intervals, with episodes lasting for 600 time steps, while for the third type it selects actions at 10 s intervals, with episodes lasting for 60 time steps. To promote co-

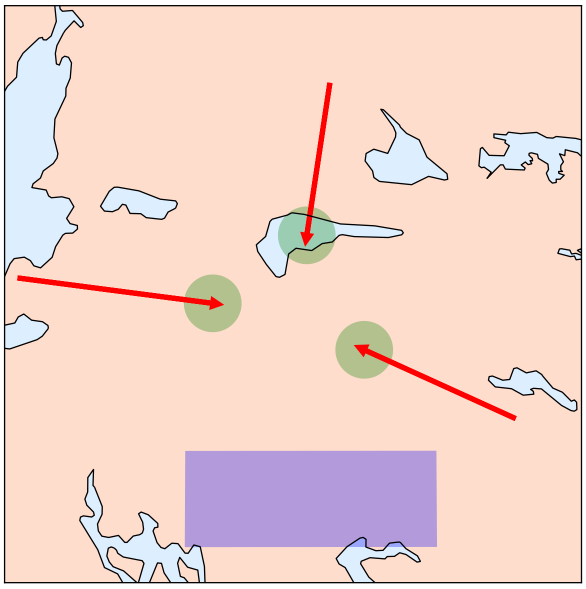

Figure 7: Rectangular spawn area of blue aircraft, and red arrows indicating attack directions of red aircraft towards high-value assets in green.

operation, the learning agents receive a shared reward defined as:

$$r_t = -\sum_{i=1}^{3} \min(\|p_{a_i} - p_{d_1}\|, \|p_{a_i} - p_{d_2}\|, \|p_{a_i} - p_{d_3}\|) \quad (4)$$

where p_{a_i} refers to the position of attacker i and p_{d_k} refers to the position of defender k.

The training progress for the low-level action spaces over 90k episodes is presented in fig. 8, and the training progress for the high-level action space over 30k episodes is presented in fig. 9. Agents that are hard-coded to always attack the same enemy are used as baselines, and their scores averaged over 1000 simulated episodes are also presented in the figures. The hard-coded baseline is strong, but not optimal, since it does not consider the initial positions of agents. To perform comparatively well, the learning agents must learn to coordinate their actions. Two of the agent types can coordinate based on only observations, while one of the agent types has the benefit of an explicit communication mechanism, provided that it can learn a protocol for coordination.

We can see that the low-level, silent controller makes fast initial improvement, but then reaches a plateau. This is because the agents must first learn to move as a team towards the protected area, before being able to learn the benefits and means of cooperation. The learning progress during the second stage of learning is quite slow, and varies among different runs, as can be seen by the increase in variance. The high-level controller, on the other hand, quickly converges to policies that perform close to the baseline. The high-level action space automatically moves the agents towards the protected area, so that agents can start learning cooperation strategies from the

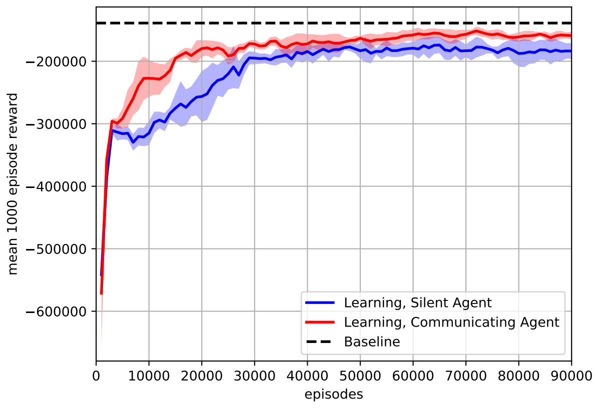

Figure 8: Mean and standard deviation for the training progress of coordinated defense with a low-level action space, for silent and communicating agents.

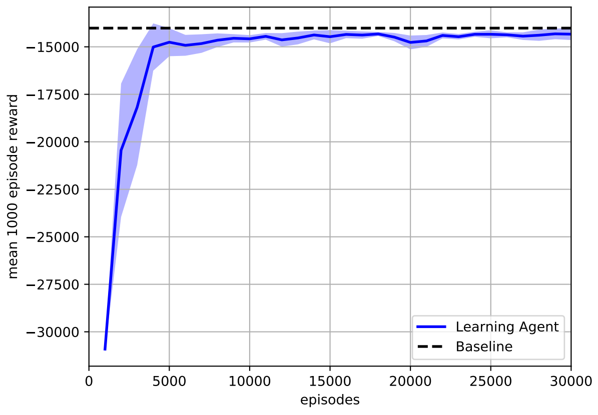

Figure 9: Mean and standard deviation for the training progress of coordinated defense with a high-level action space.

start. The low-level, communicating controller displays faster improvement, and also finds a policy that generates slightly more reward than the policy of silent agents. This indicates that explicit communication mechanisms can be valuable for efficient cooperation in air combat scenarios.

4.1.2 Coordinated Attack

In this scenario there are three targets that should be attacked by three learning agents. To carry out the task efficiently, the agents must learn to split up and attack one target each. The agents are initialized with random positions and headings in an area to the south, while the targets are initialized in random positions in a larger area to the north. The spawn areas of aircraft and targets are shown in fig. 10.

The observation space of each agent is the relative position of all other agents, as well as the targets, in a body-fixed coordinate system, for the last 4 time steps in the episode. We study the same two types of action spaces used by silent agents in 4.1.1. For the low-level action space, episodes last for 500 time steps, while for the high-level action space, episodes last for 50 time steps. To promote cooperation, the learning agents receive a shared reward defined as:

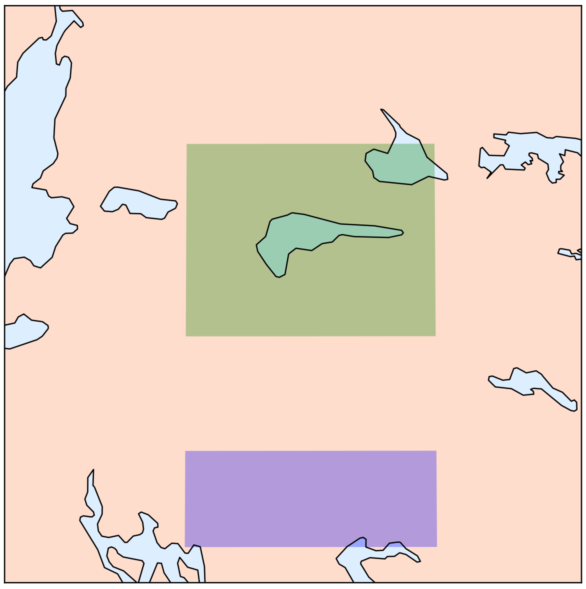

Figure 10: Rectangular spawn areas of aircraft and targets in blue and green respectively.

$$r_t = -\sum_{i=1}^{3} \min(\|p_{t_i} - p_{a_1}\|, \|p_{t_i} - p_{a_2}\|, \|p_{t_i} - p_{a_3}\|) \quad (5)$$

where p_{t_i} refers to the position of target i and p_{a_k} refers to the position of attacker k.

The agents are trained for 60k episodes. The training progress for the low-level action space is presented in fig. 11, and the training progress for the high-level action space is presented in fig. 12. Agents that are hard-coded to always attack the same target are used as baselines, and their scores averaged over 1000 simulated episodes are also presented in the figures. The hard-coded baseline is strong, but not optimal, since it does not consider the initial positions of agents. To perform comparatively well, the learning agents must learn to coordinate their actions based on only observations.

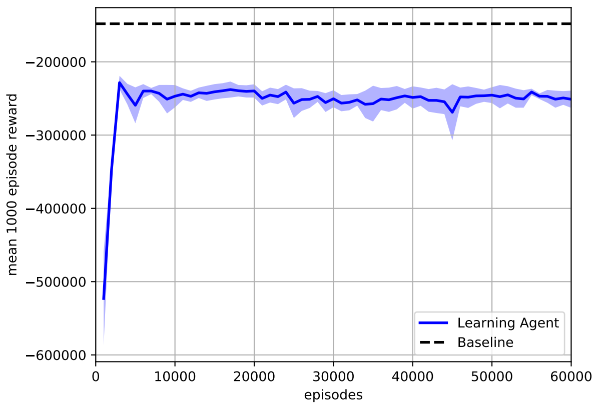

Figure 11: Mean and standard deviation for the training progress of coordinated attack with a low-level action space.

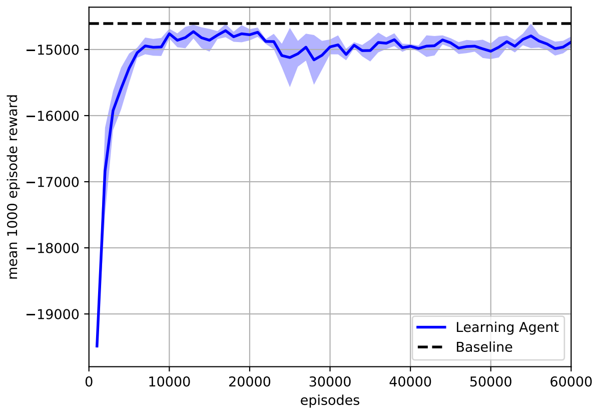

Figure 12: Mean and standard deviation for the training progress of coordinated attack with a high-level action space.

We can see that the low-level controller quickly converges to a sub-optimal policy, and then does not improve during the rest of training. Possibly further training episodes could eventually lead to an improvement of the policy. The high-level controller makes quite fast progress, but not as fast as in the experiment presented in 4.1.1. The learning also seems less stable. While some of the trained agents learn policies as good as the baseline, others struggle a bit in some episodes. This is possibly because in this scenario targets may spawn quite close to each other, which makes it difficult for the learning agents to cooperate based on observations alone.

To further study the performance of the low-level controller, and its dependence on the starting positions of agents, we conduct an additional experiment, where the aircraft are spawned in the green area in fig. 10. The training progress for this experiment over 60k episodes, with episodes lasting 300 time steps, is presented in fig. 13. We can see that the agent performs better for this scenario, since aircraft start closer to the targets, which simplifies the task of learning coordination among agents.

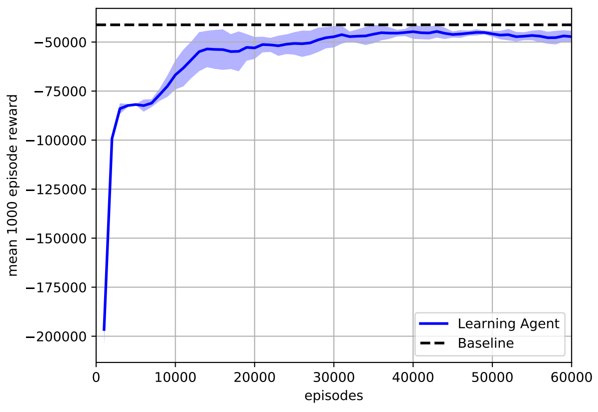

Figure 13: Mean and standard deviation for the training progress of coordinated attack with a low-level action space, when aircraft spawn close to the targets.

4.2 Risk Aware Attack

We now proceed to investigate how multi-objective reinforcement learning can be used to allow agents to learn how to prioritize among multiple conflicting objectives. We use two approaches: The outer-loop approach, where the MOMDP is converted to multiple single-objective MDPs, which are then solved with single-objective methods to produce a set of fixed policies [35], and our own approach using a single neural network conditioned on the objective priorities [37], which produces a tunable policy. We then compare the performance of the two approaches. In these experiments we use DQN as the core learning algorithm, as in the referenced papers. As previously mentioned, we use a high-fidelity simulation engine for the experiments, in contrast to the simple, low-dimensional gridworld environments studied in previous work. The policy is represented by an MLP, with 1 hidden layer with 64 neurons and the ReLU activation function. We use a learning rate of $\alpha = 10^{-4}$, a discount factor of $\gamma = 1.00$, a replay-buffer with 10^6 samples, and train using the Adam optimizer and prioritized experience replay [39].

In the studied scenario, the synthetic pilot must reach a target location in an attack mission, while avoiding enemy air defense systems. The agent must prioritize between time and safety when selecting a route. For simplicity, we place one air defense system between the agent's start position and the target. The agent always starts in the same position, with initial heading towards the target. The scenario is illustrated in fig. 16.

The observation space of the agent is the relative heading and distance to the center of the threat area, and the relative heading and distance to the target, for the last 8 time steps in the episode. Since DQN does not handle continuous actions, we must discretize the input to the controllers of the aircraft model. Thus, we define the agent's actions space as forward motion or right or left turns with a load factor of 2-4 g in discrete steps of 1 g. The agent selects actions at 1 s intervals. Each training episode is a maximum of 400 time steps long. The episode ends if the agent reaches the target. The reward vector of the MOMDP is defined as:

$$\mathbf{r}_t = [r_{goal}(t), r_{time}(t), r_{ad}(t)] \tag{6}$$

$$d_g(t) = \|p_g(t) - p_a(t)\| \tag{7}$$

$$d_{ad}(t) = \|p_{ad}(t) - p_a(t)\| \tag{8}$$

$$r_{goal}(t) = d_g(t-1) - d_g(t) \tag{9}$$

$$r_{time}(t) = -0.5 \tag{10}$$

$$r_{ad}(t) = \begin{cases} -\left(\frac{1}{10}(R_{ad} - d_{ad}(t))\right)^2 & \text{if } d_{ad}(t) \le R_{ad}; \\ 0 & \text{if } d_{ad}(t) > R_{ad}; \end{cases} \tag{11}$$

where $r_{goal}(t)$ refers to the reward for the objective of moving towards the target, $r_{time}(t)$ refers to the reward for the objective of reaching the target fast, and $r_{ad}(t)$ refers to the reward for the objective of staying out of range of the air defense system. $p_a(t)$, $p_g(t)$ and $p_{ad}(t)$ are the positions of the agent, goal and air defense system, $d_g(t)$ and $d_{ad}(t)$ are the distances from the agent to the goal and air defense system, and $R_{ad} = 20$ km is the range of the air defense system. To scalarize the vector-valued reward of the MOMDP we define the parameterized vector of priorities among objectives $\mathbf{p}_\theta = [1, \theta, \theta - 1]$, with $\theta \in [0,1]$. We then calculate a scalar reward as:

$$r_t = \mathbf{r}_t \cdot \mathbf{p} \tag{12}$$

For the tunable agent we sample θ from a uniform distribution of $[0.75, 1.00]$ before each episode, and use it as input to the agent. We train fixed policy agents for 10M time steps, while tunable policy agents are trained for 30M time steps. The training progress for fixed policy agents is presented in fig. 14, for $\theta \in \{0.75, 0.85, 0.95\}$, and the training progress for tunable agents is presented in fig. 15.

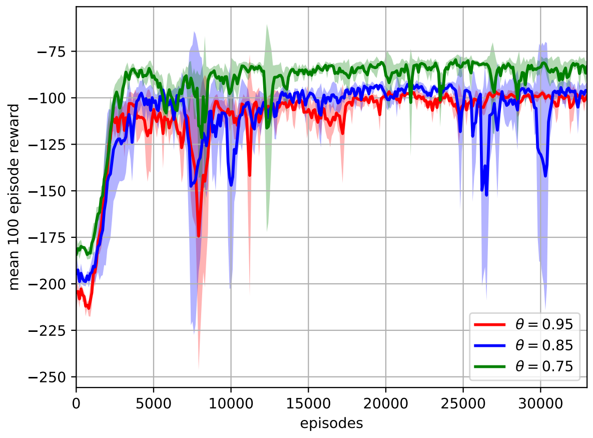

Figure 14: Training progress for a set of fixed policies with different priorities among objectives.

We can see that training is somewhat unstable, with spikes of high variance. The cause may be that small changes in policy have great effect on the accumulated reward, or that it is difficult for the agent to learn the characteristics of the reward function. It is also possible that the low frequency of the controller or the discretization of the action space has a negative effect on performance.

Three routes learned for different priorities, corresponding to high, medium and low risk exposure, are illustrated in fig. 16, for fixed policies and a tunable policy. The displayed routes are for single runs, not averaged over several runs or agents, since agents may choose to go on either side of the center of the threat area. The routes displayed for the tunable policy are from one trained agent. We can see that the tunable policy results in tighter routes around the center of the threat area, compared to those generated by the set of fixed policies. Finding an optimal route with the given reward system requires a

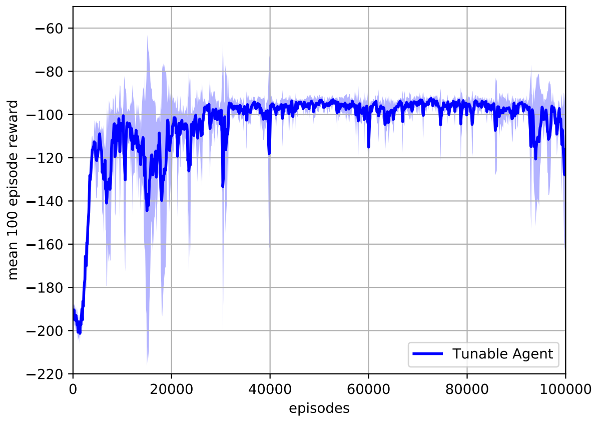

Figure 15: Training progress for single, tunable policy.

bit of tuning, and more exploration would have been required to achieve improved performance.

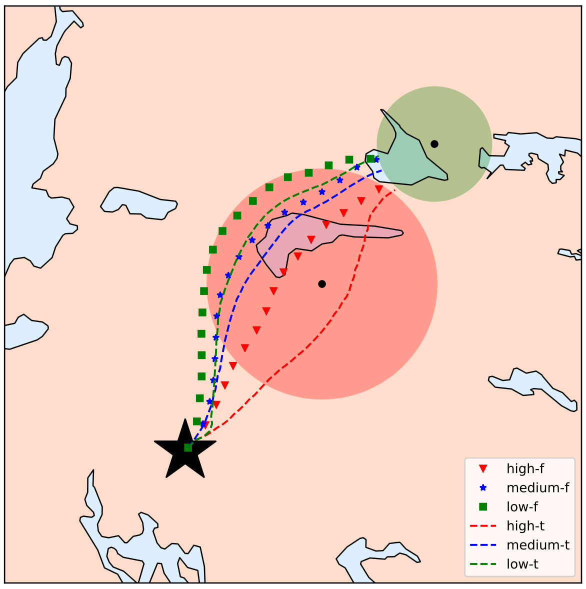

Figure 16: Learned routes to the target area with high, medium and low risk exposure, for fixed (f) and tunable (t) policies.

The mean and standard deviation for the rewards accumulated by fixed and tunable policies are presented in fig. 17. We can see that the tunable policies produce competitive results for $\theta = 0.95$ and $\theta = 0.85$, but perform worse for $\theta = 0.75$. The poor result is caused by one of the five trained agents, which fails to reach the goal for this configuration, which in turn heavily affects its accumulated reward. This also leads to high standard deviation for this case.

By extending multi-objective learning to more complex scenarios, with more objectives that must be prioritized, agents with diverse characteristics can be constructed. This can make training more interesting and stimulating, and by adjusting agents' objective preferences training contents can also be adapted to the training needs of specific trainees.

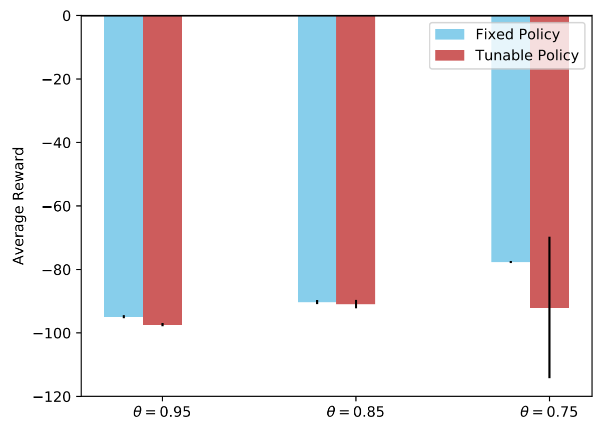

Figure 17: Mean and standard deviation for accumulated rewards for fixed and tunable policies.

5 Related Work

Over the years, there have been several attempts at using machine learning techniques for building behavior models for computer generated forces. Some approaches that have been studied are evolutionary algorithms [40–42], neural networks [43–46] and dynamic scripting [47–49], a technique originally developed for computer games. Still, the studied techniques have not been mature enough to include in commercial CGF software [9].

With the renewed interest in machine learning, sparked by e.g. AlphaGo [11,21], there have been approaches using deep reinforcement learning [50–52]. However, we are not aware of any work that studies the recent advancements in multi-agent or multi-objective deep reinforcement learning in the context of air combat simulation.

6 Conclusions

In this paper we discussed the future of air combat training, and suggested an approach for building an intelligent, synthetic trainer for fighter pilots, using machine learning techniques. We also presented results of initial experiments, which indicate that state-of-the-art algorithms can allow agents to learn team coordination as well as prioritization among conflicting objectives in simple air combat scenarios. However, we also note some challenges posed by the complexity of the air combat domain. Learning high-level tactical behavior using a low-level action space may not be the best approach. As the complexity of scenarios grows, it will become more difficult for the agent to learn efficient policies. It may get stuck in a local optimum, or perhaps not learn any reasonable policy at all. We believe that a hierarchical approach to reinforcement learning, where the problem is decomposed into a number of sub-tasks handled by a hierarchy of agents, can help tackle this problem, as indicated by the results in 4.1.1 and 4.1.2.

In future work we would like to continue to study more complex scenarios, which more closely resemble those used in operational training systems, to facilitate experiments with manned simulators and studies of human-agent interaction. We would then like to extend our study of multi-agent learning to include adversarial learning, where teams of agents compete against each-other. We would also like to combine multi-agent and multi-objective learning in an integrated architecture, using a hierarchical approach to reinforcement learning, in combination with learned models for predicting other agents' characteristics, goals and actions, to support decision making. Finally, we would like to study intelligent exploration schemes and other ways to achieve sample efficient learning in complex state and action spaces.

Acknowledgements

This work was partially supported by the Swedish Governmental Agency for Innovation Systems (NFFP7/2017-04885), and the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

References

[1] Lucian Bu, Robert Babu, Bart De Schutter, et al. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[2] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

[3] JJ Roessingh and GG Verhaaf. Training effectiveness of embedded training in a (multi-) fighter environment. Technical report, NATIONAL AEROSPACE LAB AMSTERDAM (NETHERLANDS), 2009.

[4] Ernest H Page and Roger Smith. Introduction to military training simulation: a guide for discrete event simulationists. In *1998 Winter Simulation Conference. Proceedings (Cat. No. 98CH36274)*, volume 1, pages 53–60. IEEE, 1998.

[5] Amy E Henninger, Dannie Cutts, Margaret Loper, Robert Lutz, Robert Richbourg, Randy Saunders, and Steve Swenson. Live virtual constructive architecture roadmap (lvcar) final report. *Institute for Defense Analysis*, 2008.

[6] Douglas D Hodson and Raymond R Hill. The art and science of live, virtual, and constructive simulation for test and analysis. *The Journal of Defense Modeling and Simulation*, 11(2):77–89, 2014.

[7] Jack Thorpe. Trends in modeling, simulation, & gaming: Personal observations about the past thirty years and speculation about the next ten. In *Interservice/Industry training, simulation, and education conference (I/ITSEC)*, 2010.

[8] TW van den Berg, NM de Reus, and JM Voogd. *LVC Architecture study*. Simulation Interoperability Standards Organization (SISO), 2011.

[9] Armon Toubman, Gerald Poppinga, Jan Joris Roessingh, Ming Hou, Linus Luotsinen, Rikke Amilde Løvlid, Christophe Meyer, Roel Rijken, and M Turcanık. Modeling cgf behavior with machine learning techniques: Requirements and future directions. In *Proceedings of the 2015 Interservice/Industry Training, Simulation, and Education Conference*, pages 2637–2647, 2015.

[10] Christopher Best and Benjamin Rice FLTLT. Science and technology enablers of live virtual constructive training in the air domain. *Air & Space Power Journal*, 32(4):59–73, 2018.

[11] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[12] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[13] Tetske Avontuur, Pieter Spronck, and Menno Van Zaanen. Player skill modeling in starcraft ii. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2013.

[14] Niels Justesen and Sebastian Risi. Learning macromanagement in starcraft from replays using deep learning. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 162–169. IEEE, 2017.

[15] Sid Reddy, Anca Dragan, and Sergey Levine. Where do you think you're going?: Inferring beliefs about dynamics from behavior. In *Advances in Neural Information Processing Systems*, pages 1454–1465, 2018.

[16] Neil C Rabinowitz, Frank Perbet, H Francis Song, Chiyuan Zhang, SM Eslami, and Matthew Botvinick. Machine theory of mind. *arXiv preprint arXiv:1802.07740*, 2018.

[17] Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. *arXiv preprint arXiv:1802.09640*, 2018.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[19] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[20] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[21] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[22] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4292–4301, 2018.

[23] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[24] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[25] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[26] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In *International Conference on Machine Learning*, pages 4341–4350, 2018.

[27] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[28] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[29] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[30] Christopher Amato, Girish Chowdhary, Alborz Geramifard, N Kemal Üre, and Mykel J Kochenderfer. Decentralized control of partially observable markov decision processes. In *52nd IEEE Conference on Decision and Control*, pages 2398–2405. IEEE, 2013.

[31] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[32] Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pages 271–278, 1993.

[33] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3540–3549. JMLR. org, 2017.

[34] Kristof Van Moffaert, Madalina M Drugan, and Ann Nowé. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 191–199. IEEE, 2013.

[35] Hossam Mossalam, Yannis M Assael, Diederik M Roijers, and Shimon Whiteson. Multi-objective deep reinforcement learning. *arXiv preprint arXiv:1610.02707*, 2016.

[36] Peter Vamplew, Richard Dazeley, Ewan Barker, and Andrei Kelarev. Constructing stochastic mixture policies for episodic multiobjective reinforcement learning tasks. In *Australasian Joint Conference on Artificial Intelligence*, pages 340–349. Springer, 2009.

[37] Johan Källström and Fredrik Heintz. Tunable dynamics in agent-based simulation using multi-objective reinforcement learning. In *Adaptive and Learning Agents (ALA) workshop at AAMAS*, 2019.

[38] Michele Colledanchise and Petter Ögren. *Behavior Trees in Robotics and Al: An Introduction*. CRC Press, 2018.

[39] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[40] Sandeep Mulgund, Karen Harper, Kalmanje Krishnakumar, and Greg Zacharias. Air combat tactics optimization using stochastic genetic algorithms. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, volume 4, pages 3136–3141. IEEE, 1998.

[41] Magdalena D Bugajska, Alan C Schultz, J Gregory Trafton, Shaun Gittens, and Farilee Mintz. Building adaptive computer-generated forces: The effect of increasing task reactivity on human and machine control abilities. Technical report, NAVAL RESEARCH LAB WASHINGTON DC CENTER FOR APPLIED RESEARCH IN ARTIFICIAL INTELLIGENCE, 2001.

[42] Jian Yao, Qiwang Huang, and Weiping Wang. Adaptive human behavior modeling for air combat simulation. In *2015 IEEE/ACM 19th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 100–103. IEEE, 2015.

[43] Ervin Y Rodin and S Massoud Amin. Maneuver prediction in air combat via artificial neural networks. *Computers & mathematics with applications*, 24(3):95–112, 1992.

[44] Amy E Henninger, Avelino J Gonzalez, Michael Georgiopoulos, and Ronald F DeMara. Modeling semi-automated forces with neural networks: Performance improvement through a modular approach. In *The Ninth Conference on Computer Generated Forces and Behavioral Representation Proceedings*, 2000.

[45] Teck-Hou Teng, Ah-Hwee Tan, Yuan-Sin Tan, and Adrian Yeo. Self-organizing neural networks for learning air combat maneuvers. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2012.

[46] Teck-Hou Teng, Ah-Hwee Tan, and Loo-Nin Teow. Adaptive computer-generated forces for simulator-based training. *Expert Systems with Applications*, 40(18):7341–7353, 2013.

[47] Armon Toubman, Jan Joris Roessingh, Pieter Spronck, Aske Plaat, and Jaap Van Den Herik. Dynamic scripting with team coordination in air combat simulation. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 440–449. Springer, 2014.

[48] Armon Toubman, Jan Joris Roessingh, Pieter Spronck, Aske Plaat, and Jaap van den Herik. Transfer learning of air combat behavior. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 226–231. IEEE, 2015.

[49] Armon Toubman, Jan Joris Roessingh, Pieter Spronck, Aske Plaat, and Jaap van den Herik. Rapid adaptation of air combat behaviour. In *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, pages 1791–1796. IOS Press, 2016.

[50] Roel Rijken and Armon Toubman. The future of autonomous air combat behavior. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3089–3094. IEEE, 2016.

[51] Babak Toghiani-Rizi, Farzad Kamrani, Linus J Luotsinen, and Linus Gisslén. Evaluating deep reinforcement learning for computer generated forces in ground combat simulation. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3433–3438. IEEE, 2017.

[52] Bogdan Vlahov, Eric Squires, Laura Strickland, and Charles Pippin. On developing a uav pursuit-evasion policy using reinforcement learning. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 859–864. IEEE, 2018.