Scalable Autonomous Separation Assurance with Heterogeneous Multi-Agent Reinforcement Learning

Marc Brittain, *Student Member, IEEE,*, and Peng Wei, *Member, IEEE,*

***Abstract*—In this article, a scalable autonomous separation assurance framework is proposed for high-density en route airspace sectors with heterogeneous aircraft objectives. To handle the complex dynamic decision making under uncertainty, multi-agent reinforcement learning is used in a decentralized approach with each aircraft being represented as an agent. Based on this, each agent locally solves the separation assurance problem, allowing the framework to scale to a large number of aircraft. In addition, each agent has the ability to learn the intention of the intruder aircraft, which is essential in environments with heterogeneous agents. Numerical experiments are performed in a real-time air traffic simulator. The results demonstrate that the proposed framework is able to effectively ensure the safe separation of heterogeneous agents, while also optimizing the intrinsic agent objectives in high-density en route airspace sectors. In addition, the efficiency of the proposed framework is demonstrated and shown to provide real-time decision making for separation assurance.**

***Note to Practitioners*—In commercial aviation, the workload of human air traffic controllers increases with the growth of air traffic density. Robust decision making systems to augment human air traffic controllers allows for increased air traffic without increased workload, resulting in a safer airspace environment. In addition, advanced air mobility (AAM) is concerned with low-altitude airspace operations with both human and autonomous pilots. In this environment, autonomous real-time separation assurance systems are required. Most traditional separation assurance approaches fail to handle stochastic and high-density environments, rendering them inapplicable to future high-density traditional airspace and the envisioned low-altitude AAM operations. Therefore, it is important to study decentralized approaches that can place the separation assurance problem as an intrinsic objective of each aircraft to ensure cooperation in high-density airspace. With this consideration, a scalable, decentralized autonomous separation assurance framework capable of handling heterogeneous agents is proposed in this article. This framework is able to perceive the current air traffic environment and select speed advisories to ensure safe separation requirements, while balancing intrinsic objectives such as minimizing delay. While one limitation of multi-agent reinforcement learning is the long training time, this article demonstrates how the framework also can leverage modern computing clusters to significantly reduce training time without sacrificing performance.**

***Index Terms*—Multi-Agent Reinforcement Learning, Separation Assurance, Air Traffic Management.**

M. Brittain is a Ph.D. Candidate in the Department of Aerospace Engineering, Iowa State University, Ames, IA, 50201, mwb@iastate.edu

P. Wei is an Assistant Professor in the Department of Mechanical and Aerospace Engineering, George Washington University, Washington, DC, 20052, pwei@gwu.edu

Manuscript received August 17, 2021.

I. Introduction

WITH the many recent achievements in artificial intelligence (AI), the airspace of today will look drastically different in the near future with a mix of human and autonomous pilots. This presents additional complexities for today's air traffic controllers, whose tactical decisions have only slightly evolved over the past 50 years [1]. In addition, the proposed future low-altitude operations such as Urban Air Mobility (UAM) [2], U-Space [3], and UAS Traffic Management (UTM) [4] require an autonomous air traffic control (ATC) system to provide tactical advisories to both human and autonomous pilots to ensure safe and scalable airspace operations. In a recent study, it was shown that for these low-altitude airspace operations, designing the autonomous ATC system or separation assurance system on structured airspace will be required to achieve the envisioned high traffic throughput [5]. Based on this, it is worth investigating scalable autonomous decision support frameworks to ensure safe separation between aircraft in high-density structured airspace sectors.

A. Related Work

Many techniques for managing airspace intersections and metering fixes have been designed and implemented by NASA, including the Traffic Management Advisor (TMA) [6] or Traffic Based Flow Management (TBFM), a central component of ATD-1 [7]. Conflict free time slots are determined by a centralized planner to prevent loss of separation (LOS). In addition, these traditional conflict resolution algorithms are typically based on optimization or optimal control, where a centralized controller resolves any conflicts. While these approaches are effective, it is assumed that all required aircraft and trajectory information is available to the central controller at decision time, which does not scale to high-density airspace. The central controller then generates an entire aircraft trajectory pre-departure or en route (if necessary) that is conflict free. These approaches include semidefinite programming [8], nonlinear programming [9], [10], sequential convex programming [11], [12], evolutionary techniques [13], and particle swarm optimization [14]. However, these approaches become intractable when handling high-density stochastic airspace environments due to the centralized architecture.

Collision avoidance provides the last line of defense to prevent mid air collisions (MAC) between aircraft. The planning horizon to prevent collisions is only a matter of seconds, whereas separation assurance involves a longer planning horizon. Markov Decision Processes (MDP) have successfully been applied to the collision avoidance problem as they

allow for the incorporation of a probabilistic model to handle uncertainties during flight [15]. MDP based methods can be solved offline during the pre-departure phase [16], [17], [18], [19], [20] or online during the en route phase [21], [22], [23]. Generalization becomes problematic for offline methods since the policy is designed ahead of time. Therefore, any changes in the environment en route may render the offline policy sub-optimal. In addition, the state-action space of most real-world problems is too large to represent in a discrete set of states, rendering MDP methods intractable for real-time decision making. Solutions to overcome this limitation have been proposed, such as grid-based discretization of the state-action space [24], [19] and policy compression techniques [25]. Online methods, however, overcome the limitations of offline planning by considering the current state and possible actions with more compute time. Since online methods are considering each state in real-time for decision making, these methods can adapt to changes in the environment.

While solving MDPs can be challenging, deep reinforcement learning (DRL) is a new approach that combines the advantages of offline and online methods. DRL approaches learn an approximate function through offline training that represents the policy over a continuous state-action space. This alleviates the need for state-action space discretization and can scale to large, continuous state spaces. In addition, querying DRL models is extremely efficient in comparison to approaches that solve the MDP online [21], [22]. This is due to the fact DRL methods represent a policy, mapping the state to possible actions, while also considering long-term rewards. This provides a one-step strategic decision based on the current state information, rather than making assumptions on future possible states. Theoretically, DRL has been proven effective through challenging games such as Go, Atari, Warcraft, and most recently Starcraft II with beyond human-level performance [26], [27], [28], [29]. Starcraft II is a complex, strategic game where decisions early on have long-term consequences on the outcome of the game. This demonstrates the capability of DRL methods and are worth investigating for autonomous separation assurance.

DRL in air traffic control and conflict resolution was first introduced in [30], where an AI agent was designed to mitigate conflicts and minimize the delay of aircraft reaching their metering fixes. Later, [31] demonstrated that an AI agent can effectively resolve randomly generated conflict scenarios between a pair of aircraft through vectoring maneuvers. Reference [32] developed an interactive conflict solver using reinforcement learning that leveraged human resolution maneuvers. This resulted in AI recommended maneuvers that closely aligned with human ATC behavior. More recently, [33] proposed a hybrid geometric-reinforcement learning algorithm for resolving conflicts in low-altitude airspace. While these approaches are effective for sparse airspace environments, they fail to handle state space scalability as the number of intruder aircraft increases due to the either centralized, single-agent architectures or fixed-length state vectors with a maximum number of intruder aircraft.

To handle high-density airspace environments, multi-agent formulations provide a promising solution by allowing multiple agents to work together to achieve a common objective. In the field of robotics, decentralized multi-agent reinforcement learning approaches have been explored for handling multi-robot control [34], [35], [36], [37], [38], [39], [40], [41], [42]. These approaches demonstrate that agents can effectively learn to cooperate through decentralized interactions, however, the approaches are limited to a small number of agents, rendering intractable for high-density airspace environments. Multi-agent approaches have also been investigated for conflict resolution in both structured and free airspace settings [43], [44], [21], [22], [45], [46], [47]. The key challenge in multi-agent settings is handling the non-stationarity of the environment. With multiple agents interacting in the environment, agents must communicate or learn the behavior of other agents in order to cooperate. Reference [44] introduced negotiation techniques to resolve identified conflicts in the sector. Reference [48] proposed a physics-informed deep reinforcement learning algorithm for resolving conflicts with coordination rules in traditional airspace. In [21], a message-passing based decentralized computational guidance algorithm using multi-agent Monte Carlo Tree Search (MCTS) was proposed to prevent loss of separation for UAS in an urban air mobility (UAM) setting. A computationally efficient MDP based decentralized algorithm was proposed in [22], capable of preventing LOS for UAS in unstructured airspace. Recently, [49] proposed a graph neural network approach to conflict resolution in free airspace by representing each aircraft as a node in a graph to handle scalability. In previous work [45], [46], [50], it is shown how a decentralized separation assurance framework can prevent LOS in high-density stochastic sectors by leveraging long short-term memory networks (LSTM) and attention, but this formulation only holds when all agents are homogeneous, or optimizing the same reward function.

B. Contribution of this Article

With the aforementioned considerations, this article is devoted to scalable distributed separation assurance in high-density stochastic en route sectors with heterogeneous agents. Each agent has the ability to optimize its own reward function that is unknown to other agents, or groups of agents can share the same reward function. The agents also have the ability to learn the intentions of other agents, which is critical to ensure cooperation in this environment. This provides a framework that is able to accommodate the requirements of a mixed autonomy airspace environment with competing companies where autonomy models may be proprietary. In other words, this framework is flexible enough to handle an arbitrary number of reward functions. By leveraging the work of [50], this framework also benefits from modern compute cluster environments, which significantly reduces the training time and intruder aircraft scalability, providing a promising solution to autonomous separation assurance in traditional and low-altitude airspace operations.

The distinctive features of the proposed separation assurance framework are three-fold. First, the proposed multi-agent autonomous separation assurance (MAASA) framework is efficient and capable of scaling to high-density airspace

environments. Second, the framework is able to learn the intention of agents without explicitly knowing their objectives, which more closely resembles encounters that will occur in near future autonomous airspace operations. Third, while this article focuses on en route structured airspace, the generality of the framework allows it to be extensible to terminal area and 3-dimensional airspace operations.

The remainder of this article is organized as follows. The problem is first formulated in Section II. The MAASA framework is proposed in Section III. Numerical experiments are described in Section IV, and Section V concludes this article.

II. Problem Description

Separation assurance involves providing advisories to aircraft to prevent a loss of separation (LOS) event with other aircraft in-trail, at intersections, and at metering fixes. The loss of separation threshold, d^{LOS}, defines a safety radius where operations within the threshold become increasingly dangerous. Violating the loss of separation threshold may result in collisions or near mid air collisions (NMACs) between aircraft, which also results in drastic maneuvers from the aircraft. Therefore, it is an essential task for air traffic controllers to maintain safe aircraft separation in the airspace. In addition, the separation assurance task also involves ensuring that aircraft meet the required time of arrival (RTA) for metering fixes. These RTAs allow controllers to prevent arrival delays by ensuring aircraft are on schedule for airport arrivals. Any given aircraft in the environment is referred to as an ownship, with the other aircraft associated with the ownship referred to as the intruder aircraft. In this way, each ownship will have its own associated intruder aircraft from the point of view of the ownship.

While the separation assurance task is performed by human air traffic controllers (ATC), with growing air traffic operations, this task pushes the limit of human cognitive decision making. Therefore, by introducing an autonomous separation assurance framework, human ATCs can provide a supervisory role over the autonomous system and intervene if necessary, reducing their overall workload in high-density airspace sectors.

In addition, it is also essential to consider the future operations in low-altitude airspace including UAM and UTM, with regard to how these operations will integrate with traditional airspace operations. In this low-altitude airspace environment, instead of a centralized government authority like the FAA, companies will operate their own autonomous aircraft with potentially proprietary software. This provides additional complexity since agents will need infer the behavior of other aircraft to form cooperation since there is no centralized authority and the parameters of the proprietary software may be unknown.

III. Multi-Agent Autonomous Separation Assurance

Given the aforementioned problem, in this section the multi-agent autonomous separation assurance (MAASA) framework is constructed.

A. Multi-Agent Reinforcement Learning

Consider the problem of N aircraft operating in a given airspace sector at time t. For each ownship, there will be an associated $N-1$ intruder aircraft. However, since the airspace environment is dynamic, with aircraft constantly entering and exiting sectors, the value of N is not fixed over time. This results in a variable number of intruder aircraft that the ownship must sense at each time step to maintain safe separation. Multi-agent reinforcement learning provides a solution approach to the variable aircraft separation assurance problem. Each aircraft is represented as an agent and has an associated state space S (information the agent can sense), action space A (actions the agent can implement in the environment), and reward function R. At each time step t, the agent senses a state s_t and selects an action a_t. The environment is then updated as a result of the state-action tuple (s_t, a_t) to s_{t+1} and an associated reward r_t is received. How the environment updates from s_t to s_{t+1} is based upon the dynamics of the environment, which are often unknown. In this way, the separation assurance problem is decentralized with local agent decision making, rather than a centralized approach, which does not scale with the number of agents. Since this article is extending the work [50], this article uses a similar formulation of state space, action space, and reward function to ensure interoperability. The formulation is now briefly discussed.

1) State Space: The state space, S defines the information the agent requires to effectively make decisions. In this article, it is assumed that each ownship can sense the position and dynamics of the intruder aircraft. The state space for the ownship and intruder aircraft is then defined as

$$s_t^o = (x^{(o)}, y^{(o)}, d_{\text{goal}}^{(o)}, v^{(o)}, a^{(o)}, hdg^{(o)}, d^{\text{LOS}}, m^{(o)}),$$

$$h_t^o(i) = (x^{(i)}, y^{(i)}, d_{\text{goal}}^{(i)}, v^{(i)}, a^{(i)}, hdg^{(i)}, m^{(i)}, d_o^{(i)}, d_{\text{int}}^{(o)}, d_{\text{int}}^{(i)}),$$

where s_t^o is the state of the ownship at time t and $h_t^o(i)$ is the state for intruder i at time t. The elements of the state space for the ownship include the easting and northing location in Universal Transverse Mercator (UTM) coordinates $(x^{(o)}, y^{(o)})$, the distance to the goal, or distance to the sector exit ($d_{\text{goal}}^{(o)}$), aircraft speed ($v^{(o)}$), aircraft acceleration ($a^{(o)}$), aircraft heading ($hdg^{(o)}$), the loss of separation threshold (d^{LOS}), and an identifier for which model the ownship belongs to ($m^{(o)}$). The state for the intruder aircraft contains similar elements, except since the LOS threshold is in the ownship state space, there is no need to repeat this information in the intruder state. The three additional elements in the intruder state include the straight-line distance from the ownship to the intruder ($d_o^{(i)}$), distance from the ownship to the intersection ($d_{\text{int}}^{(o)}$), and the distance from the intruder to the intersection ($d_{\text{int}}^{(i)}$). UTM coordinates were selected over latitude and longitude, because they provide a way to represent the location in unit meters. Since neural networks require normalized inputs, this provides a more standard normalization unit irrelevant to the size and location of the en route sector, whereas latitude and longitude normalization may introduce approximation errors. The term intersection as it is used in this article can refer

to a crossing point between two air routes, or a merging point of air routes. The reasoning behind including these terms in the intruder state space and not in the ownship state is to create a fixed size state space representation. The term $d_{\text{int}}^{(o)}$ represents the distance from the ownship to the intersection, however for a given air route there may be many intersections and different air routes may have a different number of intersections. If this information was included in s_t^o, the result would be a state that does not scale with the number of intersections in the air route as there would need to be a term for all possible intersections M, $d_{\text{int}^{1:M}}^{(o)}$. However, this issue can be resolved by recognizing that an intersection is a potential conflict point between two aircraft. Therefore, only the intersection information associated with a given intruder needs to be considered. By including this information in the variable-length intruder state, h_t^o, this formulation is invariant to the number of intersections for the ownship.

2) Action Space: Actions for the agent reflect horizontal in-trail speed advisories, with a decision step of 12 seconds. The decision step parameter can be modified based on the application, as surveillance requirements may vary based on the location of operations (e.g., near-terminal operations or trans-oceanic operations). The action space is defined as

$$a_t = \{a_-, 0, a_+\},$$

where a_- is to decelerate (decrease speed), 0 is no acceleration (hold current speed), and a_+ is acceleration (increase speed). The magnitude of acceleration is dependent on the performance envelope for a given aircraft type. Using acceleration is analogous to a real-world air traffic controller setting a new desired speed for the aircraft. Setting a new desired speed will result in an acceleration change which is reflected in the action space. Given the performance envelope of the aircraft type is considered, the selected actions that would result in speeds outside of the aircraft's performance envelope have no effect. This provides a more realistic scenario, as aircraft have minimum and maximum operating speeds.

3) Reward Function: The reward function needs to be carefully designed to reflect the objective of separation assurance. In this article, the reward function from [50] is extended to also include a time step penalty. This encourages agents to move through airspace quickly while maintaining safe separation. The reward function for the state and action is defined as

$$R(s) = \begin{cases} -1 & \text{if } d_o^c < d^{\text{LOS}} \\ -\alpha + \delta \cdot d_o^c & \text{if } d^{\text{LOS}} \leq d_o^c < 20 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$R(a) = \begin{cases} 0 & \text{if } a = \text{`Hold'} \\ -\psi & \text{otherwise} \end{cases}, \tag{2}$$

where $R(s)$ is the reward for a given state and $R(a)$ is the reward for a given action. The total reward for a given time step can then be combined as

$$R(s, a) = R(s) + R(a) + \nu. \tag{3}$$

In $R(s)$, the term d_o^c represents the distance from the ownship to the closest intruder aircraft. Therefore, the first inequality captures the loss of separation penalty of -1, which is the worst possible penalty. The middle inequality in $R(s)$ represents a buffer zone where the agent attempts to maximize the minimum separation between the ownship and the closest intruder. The values of α and δ are scaling constants to ensure that the value of the reward is between $(0, -1)$. The penalty linearly increases as d_o^c decreases, encouraging the ownship to maximize the distance from the closest intruder when possible. $R(a)$ penalizes speed change advisories that are not holding actions. Consecutive speed change advisories are undesired and may negatively impact fuel consumption during a flight. By incorporating this penalty, the agent must learn to maintain safe separation, while also minimizing the number of speed advisories. Finally, in the total reward function, ν is a time step penalty that encourages agents to move through the airspace sector as quickly as possible. This encourages increased throughput while also achieving the aforementioned objectives.

4) State Termination: The agents continue to operate in the environment until a terminal state is reached. There are two ways a terminal state is reached in the environment. First, the agent can safely navigate through the sector without violating safe separation requirements. This is referred to as a *goal*. Second, the agent can violate the safe separation requirements. One entire simulation run, hereafter referred to as an episode, completes when all agents have reached their terminal state,

$$N_{\text{aircraft}} = 0.$$

B. Heterogeneous Reinforcement Learning

While [50] introduced a general framework for decentralized separation assurance with multi-agent reinforcement learning, one key limitation is the assumption that all agents follow the same reward function (or model). In future traditional airspace, and more specifically in low-altitude airspace operations, the objectives of other aircraft may be unknown. In these cases, rather than having homogeneous agents optimizing the same reward function, the environment now involves heterogeneous agents optimizing different reward functions. This presents a challenging environment where companies may have their own optimization scheme that differs in the objective of others. To illustrate, consider two companies, company A and company B, optimizing the reward function in (1) and (2). Company A decides to select $\psi = 0$ and company B decides to select $\psi = 0.01$. The result of this slight difference will be the following policy. The policy for company A will not penalize alternating speed advisories, while the policy for company B will be more conservative, attempting to minimize the number of speed advisories. This violates the cooperation assumption in [50], since agents are no longer optimizing the same reward function and their polices may differ.

In this article, a mechanism for learning the intruder aircraft's intent is proposed to alleviate the requirement that agents need to optimize the same reward function. Suppose

there are M reward functions and the agents are free to select any of the M functions. If the number of agents, N is larger than M, then agents can group into the same reward function. Each reward function will have an associated neural network model with weights θ that is optimized to learn a policy that reflects the behavior of the reward function. To alleviate the constraint of a single reward function, a second model with weights, ϕ is introduced to predict the action distribution for the intruder aircraft. Each intruder aircraft may belong to any of the M reward functions, so by predicting their intended action, the ownship is able to select an action that cooperates with the intruder's policy, as all agents are still required to maintain safe separation.

1) Proximal Space Optimization: The reinforcement learning algorithm used in [50] was proximal policy optimization (PPO) [51] with general advantage estimation [52], and this article leverages this approach. Proximal policy optimization is a policy based reinforcement learning algorithm where the basic idea is to learn a parameterized policy (a mapping from a given state to a probability distribution over the actions) to maximize the cumulative reward J, by following the gradient of J with respect to the policy parameter. Policy based algorithms have advantages over value based reinforcement learning algorithms, as stochastic policies can be learned. A neural network is used to approximate both the policy (actor) and the value function (critic), and it is often common to use shared layers of a single neural network for both the actor and critic. This neural network is optimizing the following loss functions

$$L_\pi(\theta) = -E_t[\min(\zeta_t(\theta)\cdot A_t, \text{clip}(\zeta_t(\theta), 1-\epsilon, 1+\epsilon)\cdot A_t)] - \beta \cdot H(\pi(s_t)) \quad (4)$$

$$L_v = A_t^2, \quad (5)$$

where L_π is the policy loss and L_v is the value loss. ϵ is a hyperparameter that bounds the policy changing ratio $\zeta_t(\theta)$. The second term in Equation (4), $\beta \cdot H(\pi(s_t))$ is an entropy regularization term to encourage exploration by preventing early convergence to sub-optimal polices. H is the entropy of the policy distribution and the hyperparameter β controls the strength of the entropy regularization term. The advantage function A_t in Equations (4) and (5) provides a measure of which action is better or worse than the current policy's behavior. In this way, actions with high advantage will be further reinforced into the policy, with actions resulting in low advantage discouraged. The generalized advantage estimator GAE(γ, λ) [52] provides a way to approximate A_t, which is defined as the exponentially-weighted average of the k-step advantage estimators

$$A_t = (1-\lambda)(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + \cdots) \quad (6)$$

where

$$\begin{aligned} \hat{A}_t^{(1)} &= -V(s_t) + r_t + \gamma V(s_{t+1}) \\ \hat{A}_t^{(2)} &= -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 V(s_{t+2}) \\ \hat{A}_t^{(3)} &= -V(s_t) + r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 V(s_{t+3}) \\ &\cdots \\ \hat{A}_t^{(k)} &= -V(s_t) + \textstyle\sum_{i=t}^{t+k-1} \gamma^{i-t} r_i + \gamma^k V(s_{t+k}). \end{aligned} \quad (7)$$

2) Learning Intruder's Intent: In a real world setting, agents would be unable to directly observe the action distribution of the intruder aircraft; the agents would only be able to observe their resulting action. In this case, since the intention learning model with weights ϕ is learning the action distribution of the intruder aircraft, a natural choice for the loss function is the cross entropy. This way, the model is trained to minimize the difference between the predicted distribution and the true resulting action. Throughout training, the predicted distribution will approach the true action distribution of the intruder aircraft through observation of the intruder's action. The cross entropy loss is formulated as

$$L_{\text{CE}} = -\sum_{i=1}^{C} t_i \cdot \log(p_i), \quad (8)$$

where C is the number of actions, t_i is the true action for a given intruder (one-hot encoded), and p_i is the predicted action distribution.

There are two ways to train the intention learning model to optimize the loss function. The first is by updating the model directly before updating the actor-critic model in an online manner as in [53]. The second is by using a shared model architecture to jointly train the actor, the critic, and the intention learning model ($\theta = \phi$). Through initial architecture exploration, it was found that jointly training the models resulted in overall faster learning and better performance, so this article focuses on a joint training formulation. Therefore, the total loss function for the MAASA framework is defined as

$$L = L_\pi + L_v + L_{\text{CE}}. \quad (9)$$

This provides an intuitive extension to the shared layer loss function of [50]. The loss in [50] can be recovered by simply setting $L_{\text{CE}} = 0$. This provides a model that is applicable to both homogeneous and heterogeneous agents.

C. Architecture

The MAASA architecture is illustrated in Fig. 1, with the intention learning model illustrated in Fig. 2. The intention learning model is implemented architecturally as a neural network and optimized via the cross entropy loss as discussed in Section III.B.2. The input corresponds to the preprocessed intruder state information h_t^e, which is then processed through three fully connected layers with 128 nodes. The output is a fully connected layer with 3 nodes (1 for each action) with a softmax activation applied to provide the predicted action distribution $\hat{a}_t$. The intention learning model is incorporated into the MAASA framework as shown in Fig 1. In the MAASA

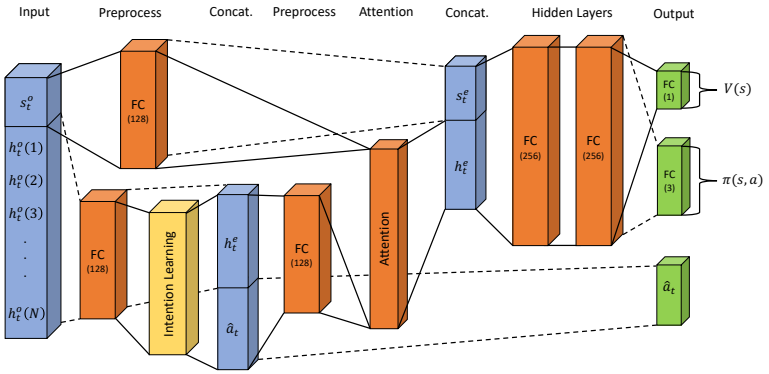


Fig. 1: MAASA neural network architecture. The input is the ownship state and the variable number intruder state information. The output is the value, policy, and predicted action distribution from the intention learning model (shown in yellow).

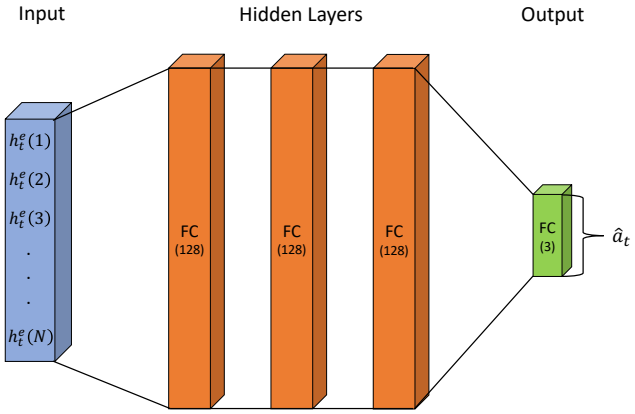


Fig. 2: MAASA intention learning neural network architecture. The input is the preprocessed intruder state information and the output is the predicted action distribution for each intruder.

framework, the ownship and intruder state information (s_t^o and h_t^o, respectively) is first preprocessed through two fully connected layers with 128 nodes to provide a consistent feature dimension. Then, the intruder information is processed through the intention learning model (Fig. 2) to extract the predicted action distribution. This predicted action distribution for each intruder is concatenated with the already preprocessed intruder state information. This is an important step, because now this concatenated vector contains the intruder state information, along with the predicted action distribution, providing more information for the ownship to make a decision. This concatenated vector is processed through a fully connected layer of 128 nodes to ensure that the concatenated vector has a consistent feature dimension with the preprocessed ownship state information. This is required since concatenating the action distribution results in the dimension of the vector increasing from 128 to 131 (3 actions) and the attention network requires consistent feature dimensions. From there, the ownship and intruder information is processed through an attention network as defined in [50] to encode the variable number of intruder aircraft into a fixed-length vector. The fixed-length vector is then concatenated with the preprocessed ownship information before being sent through two fully connected layers with 256 nodes. The output of the framework is the value, the policy, and the action prediction. Unless otherwise noted, each fully connected was followed by a Leaky ReLU activation function.

IV. Experiments

To evaluate the performance of the proposed MAASA framework, the open-source BlueSky [54] air traffic simulation environment was used to develop several case studies. The BlueSky simulator provides realistic, fast-time simulations with real-world aircraft performance data. In addition, leveraging open-source environments provides equal opportunity for researchers and industry to baseline implementations. While the environment is not designed for reinforcement learning, [50] introduced a scalable reinforcement learning extension to BlueSky that allows for integration with modern cluster computing environments. This approach uses the Ray [55] python module to allow for parallel CPU threads to simultaneously interact with their own copies of the BlueSky environment, greatly increasing the number of simulations that can be executed. This was found to not only decrease training time to hours instead of days, but also increase performance. All

TABLE I: Finalized hyperparameters for MAASA.

Parameter	Value
Learning Rate	0.0001
GAE Discount Factor γ	0.99
GAE Discount Factor λ	0.95
PPO Ratio Bound ϵ	0.4
Entropy Coefficient β	0.0001
Parallel Workers	75
Leaky Relu Alpha	0.2

experiments were run on a AMD Ryzen Threadripper 2950x (16 cores, 32 threads) workstation with a Nvidia RTX 2080 TI GPU (12 GB) and 128 GB RAM[1]. MAASA parameters were selected through hyperparameter tuning. Table I provides the final parameter values used in the numerical experiments.

A. Experimental Setup

1) Baseline: The baseline introduced in this article to compare performance is the D2MAV-A framework [50]. D2MAV-A is considered the state-of-the-art in distributed separation assurance with reinforcement learning and is shown to be capable of minimizing loss of separation events for homogeneous agents. The main difference between the MAASA framework and the D2MAV-A is the intention learning model. The D2MAV-A does not have the ability to predict the action distribution of intruder aircraft and is restricted under the assumption that all aircraft follow the same policy, whereas the MAASA framework removes this assumption by introducing the intention learning model to predict the action distribution for intruder aircraft. This article also provides the result of a random agent (when applicable) to compare the worst-case performance.

2) Case Studies: To provide a fair comparison with the D2MAV-A framework, the three case studies introduced in [50] (case study A, B, and C) are used, along with a new case study D. An illustration of case study D is shown in Fig 3. The case studies represent en route sectors of varying number of route and intersections. The characteristics of each case study are provided in Table II. In each episode (simulation run), 50 aircraft are sent through the sector following a uniform inter-arrival distribution of three to six minutes. This creates a high-density airspace environment where the agents must learn to generalize to unforeseen aircraft orientations, given the stochastic inter-arrival times. Fig. 4 displays the number of aircraft in each case study over time for a given episode. Each case study provides a scenario where over 10 agents are selecting speed advisories at the same time, which is infeasible for a human. Case study D provides an extremely high-density sector where close to 50 aircraft are selecting a speed advisory at the same time. Heterogeneity in the airspace is introduced through five different reward functions, where the agents are uniformly assigned to a neural network model (θ_i) that is optimizing the objectives of one of the reward functions. Therefore, there will be a total of five (neural network) models corresponding to each of the reward functions. The parameters of the reward functions are provided in Table III.

[1]Code will be made available at https://github.com/marcbrittain

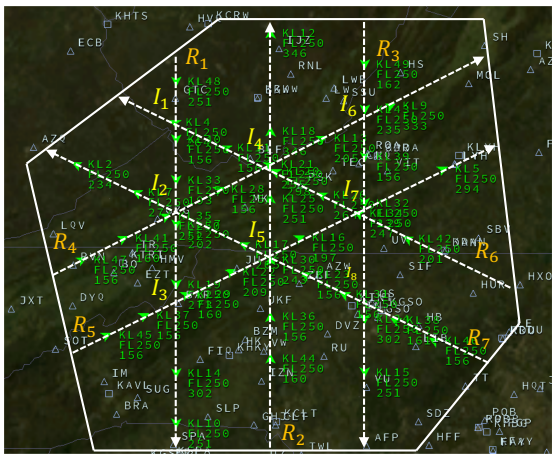


Fig. 3: Case study D en route sector.

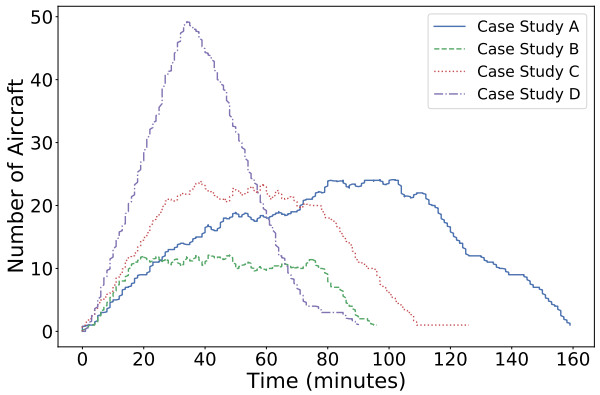


Fig. 4: Number of aircraft in the sector over time for each case study.

TABLE II: Characteristics of the en route case study sectors.

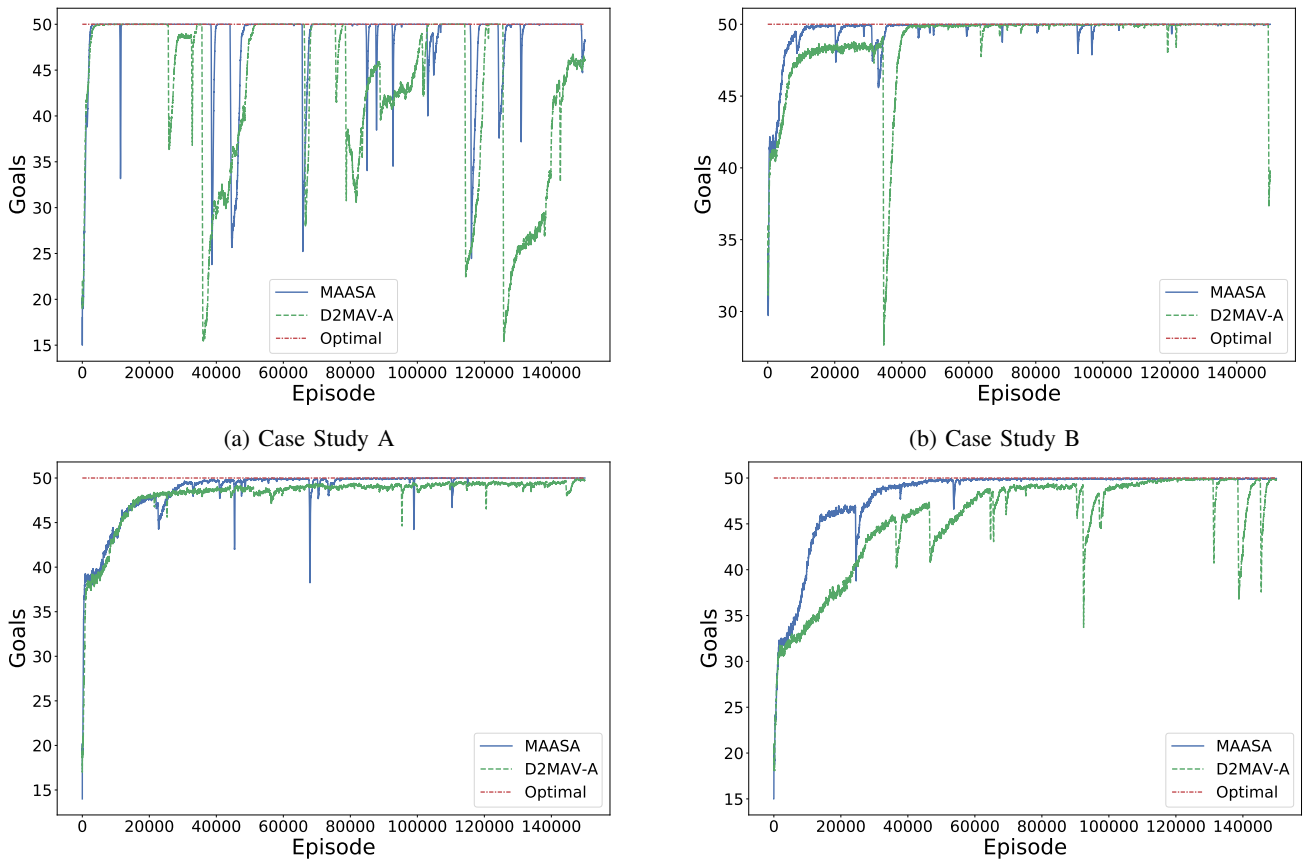
Case Study	Routes	Intersections
A	2	1
B	3	2
C	3	3
D	7	8

TABLE III: Parameters values used for each reward function.

Parameter	R_1	R_2	R_3	R_4	R_5
Reward Coefficient α	0.1	0.1	0.1	0.1	0.1
Reward Coefficient δ	0.005	0.005	0.005	0.005	0.005
Reward Coefficient ψ	0.001	0	0.001	0.002	0
Reward Coefficient ν	0	0.001	0.001	0	0.002

B. Experimental Results

In this section, the performance of the MAASA framework is evaluated on case studies through various numerical experiments. For all experiments, the agents were allowed to train for 150k episodes, with 50 Airbus A320 aircraft in each episode. After training, a testing phase is then performed for 200 new episodes to evaluate the performance of the frameworks. Therefore, the optimal solution is 50 goals, or 50 aircraft exit the sector without violating loss of separation. Unless otherwise noted, three reward functions (R_1, R_2, and



(a) Case Study A

(b) Case Study B

(c) Case Study C

(d) Case Study D

Fig. 5: Learning curves smoothed with 200-episode rolling average for clarity.

R_3 from Table III) are used in all experiments to create the heterogeneous airspace.

1) Learning Efficiency: It is important to understand how efficient the framework is regarding model training. Giving that computational resources can be expensive, minimizing compute time while maximizing performance is an important attribute for a framework to have. This experiment involved evaluating the number of episodes until convergence during training, where convergence is defined as the first time the optimal score (50 goals achieved) is obtained over a 150-episode rolling average. Table IV provides the results of this experiment. For case studies A and B, the MAASA framework is able to converge in much fewer episodes in comparison to the D2MAV-A framework. In addition, for case studies C and D, the MAASA framework converged in less than 100k episodes, with the D2MAV-A framework failing to converge over the 150k training episodes. The learning curves throughout training are shown in Fig. 5. It can be seen in Fig. 5 that for each case study, the MAASA framework learns faster than the D2MAV-A framework. In case study A, both frameworks quickly converge to the optimal score, with additional exploration resulting in spikes in the learning curve. Case studies B, C, and D follow a similar trend where the MAASA framework quickly diverges from the D2MAV-A framework, achieving better performance in fewer training episodes.

TABLE IV: Number of training episodes until convergence.

Case Study	MAASA	D2MAV-A
A	**3110**	5225
B	**28278**	67172
C	**81539**	-
D	**81898**	-

When observing Table IV, it is important to realize why the MAASA framework is able to learn more efficiently than the D2MAV-A framework. As mentioned earlier, non-stationarity is a key challenge in multi-agent reinforcement learning. With each agent learning and updating their own individual polices, the agents can enter an endless cycle of adapting to the other agent's polices, losing sight of their individual objectives [56]. Non-stationarity becomes even more pronounced in heterogeneous environments where the agents may now belong to different learned models and do not have a sense of the objectives of the other agents. This breaks the Markov assumption that governs many single agent RL algorithms. Without a mechanism for the agents to explain why the other agents are acting a certain way, the agents may never converge to the optimal behavior [56]. By learning the intention of the intruder aircraft, each ownship is able to obtain an internal estimate of how the intruder aircraft is going to behave, reducing the non-stationarity in the learning

TABLE V: Performance of the policy tested for 200 independent episodes.

	Case Study A		Case Study B		Case Study C		Case Study D	
Framework	Mean	Median	Mean	Median	Mean	Median	Mean	Median
MAASA	**49.75 ± 0.719**	**50**	**50.0 ± 0.0**	**50**	**50.0 ± 0.0**	**50**	**49.92 ± 0.44**	**50**
D2MAV-A	47.52 ± 2.21	48	49.96 ± 0.28	50	49.84 ± 0.578	50	49.775 ± 0.689	50
Random	19.6 ± 3.54	20	32.455 ± 4.31	32	19.62 ± 3.33	20	19.19 ± 3.21	20

TABLE VI: Performance of the policy tested for 200 independent episodes with varying number of models.

	1 Model			3 Models			5 Models		
Framework	Mean	Median	LOS Events	Mean	Median	LOS Events	Mean	Median	LOS Events
MAASA	**50.0 ± 0.0**	**50**	**0**	**50.0 ± 0.0**	**50**	**0**	**49.98 ± 0.199**	**50**	**4**
D2MAV-A	50.0 ± 0.0	50	0	49.84 ± 0.578	50	32	49.86 ± 0.51	50	28

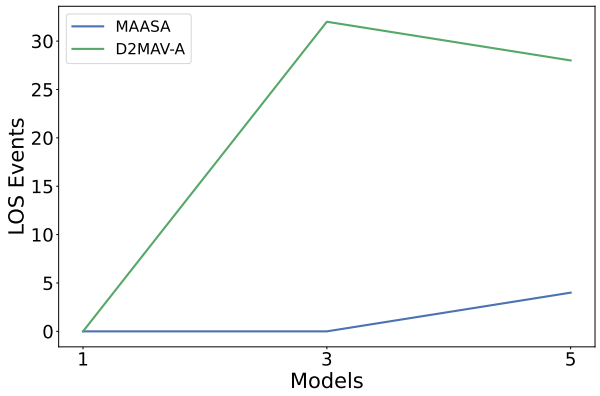


Fig. 6: Total number of LOS events during the testing phase.

process. In contrast, without the intention learning model the D2MAV-A framework takes significantly longer to reach similar performance to MAASA (case study B), or never converges (case study C and case study D).

The important takeaway from this result is that convergence can be achieved in minimal compute time, as training for 100k episodes only takes approximately 10 hours of wall-clock training time. Given that the MAASA framework converged in **3110** episodes for case study A, this means that in approximately 30 minutes a converged policy can be obtained.

2) Policy Evaluation: To evaluate the overall performance of the MAASA framework, the best model weights from training are extracted and tested on 200 new episodes[2]. The mean and median episode scores are recorded in Table V. Given that 50 aircraft are sent through the sector, the optimal performance is 50, representing all aircraft exited the sector without violating the loss of separation threshold. It can be seen that across all case studies, the MAASA framework results in a significant improvement over the D2MAV-A framework. While the MAASA framework achieved a score above 49.7 on all case studies, the D2MAV-A framework performed poorly on case Study A and achieved a lower score across all case studies. This shows how important learning the intention of the intruder aircraft is to obtain good performance.

[2]A video of the converged policy for case study D can be found at https://youtu.be/Udj4a7uLNXE

3) Heterogeneous Airspace: To further demonstrate the performance of the MAASA framework in heterogeneous airspace, the number of reward functions is varied from one to five and evaluated on case study C. Each reward function has a corresponding model that the agents are randomly assigned to upon entering the sector, diversifying the objectives the agents are optimizing. Therefore, increasing the number of models is analogous to increasing the heterogeneity of the airspace environment. One model represents a homogeneous airspace (all aircraft obeying the same model), three models represents medium heterogeneous airspace, and five models represents high heterogeneous airspace. The evaluation criteria follows the same method as described earlier, where the agents train for 150k episodes and then test on 200 new episodes. The results from the testing phase are shown in Fig. 6 and Table VI. It can be seen from Table VI that the MAASA framework leads to consistently better performance in comparison to the D2MAV-A framework, even as the heterogeneity of the airspace increases.

While the mean scores in Table VI are relatively close between MAASA and D2MAV-A, there are significant differences in safety between these two frameworks. As displayed in Fig. 6, during the testing phase, it can be observed that the D2MAV-A framework results in more LOS events compared to the MAASA framework. When the airspace is homogeneous, the number of LOS events is equivalent between the D2MAV-A and MAASA frameworks, which is expected given the D2MAV-A framework is operating on the assumption that all aircraft obey the same model. Once heterogeneity is introduced into the airspace through three and five different models, the D2MAV-A framework results in 32 and 28 LOS events, respectively. In contrast, the MAASA framework has 0 LOS events when there are three models and 4 LOS events when there are five models. This equates to a 86% decrease in LOS events when there are five models (high airspace heterogeneity). In safety critical applications such as air transportation, minimizing the number of LOS events is essential for preventing any potential mid-air collisions. Therefore, the MAASA framework is shown to be a safer option in comparison to the D2MAV-A framework.

Performance degradation as seen in Table VI is expected as the number of models increases, given the conservative training time. Further performance gains can be expected given

TABLE VII: Performance of the policy tested for 200 independent episodes with varying interarrival distribution.

	3-6 Min.		3-8 Min.		3-10 Min.	
Framework	Mean	Median	Mean	Median	Mean	Median
MAASA	**50.0 ± 0.0**	**50**	**50.0 ± 0.0**	**50**	**50.0 ± 0.0**	**50**
D2MAV-A	49.84 ± 0.578	50	50.0 ± 0.0	50	50.0 ± 0.0	50

more training episodes, however, it is important to demonstrate the relative performance under constrained compute resources. In this case, the MAASA framework shows to be a better choice of framework when handling heterogeneous airspace.

4) Sensitivity Analysis: The three to six minute uniform interarrival distribution provides a high-density airspace environment that is more challenging in comparison to the density of today's airspace. In today's airspace, each route has its own interarrival distribution that can vary from minutes to tens of minutes and hours. Given that the experiments assume that each route follows the same distribution, this acts as a stress test to view the performance of the framework under max airspace capacity. Therefore, it is important to understand the sensitivity of the MAASA framework under different interarrival distributions. The MAASA framework is evaluated with three interarrival distributions as shown in Table VII for case study C, following the same evaluation procedure as described earlier. The 3-10 minute interarrival distribution represents a low-density airspace, 3-8 minutes represents medium-density airspace, and 3-6 minutes represents high-density airspace. As seen in Table VII, the performance of the MAASA and D2MAV-A frameworks are equivalent in low to medium-density airspace, but differ in high-density airspace where MAASA outperforms D2MAV-A. In low to medium-density airspace, it is likely that the aircraft may be sufficiently separated so that the impact of heterogeneity is not as pronounced. When operating in high-density airspace, MAASA is the best option given it obtains the optimal score of 50 (no LOS events), while the D2MAV-A framework achieves a score of 49.84 (LOS events occurred).

5) Intention Learning Prediction: To evaluate the performance of the intention learning model, the predicted intruder intent is compared with the true action of the intruder aircraft to obtain the accuracy of the prediction. This provides a more comprehensible metric, in contrast to the cross entropy loss itself. To calculate the accuracy, the true action distribution of the intruder is recorded, along with the predicted action distribution. The action with the maximum probability is selected for both distributions and compared to obtain the accuracy. The result of this experiment is shown in Table VIII. It can be seen that in case study A, B, C, the accuracy of the intention learning model was greater than 90%, with the accuracy slightly below on case study D at 89%. 90% accuracy demonstrates that the intention learning model is accurately predicting the actions of the intruder aircraft, and by having this information, the overall performance of the MAASA framework is improved. Given that only 150k training episodes were used, it is suspected that with more training better performance can be achieved.

While the accuracy provides a more interpretable metric for performance, it is important to note that since each intruder is also randomly selecting their action, actions with equal probability may not be properly reflected by this metric. For example, if the true action distribution for an intruder is $[0.33, 0.33, 0.34]$ and the predicted action distribution is $[0.34, 0.33, 0.33]$, the accuracy metric would assign the true action as a_+ (third element) and the predicted action as a_- (first element), resulting in an incorrect prediction. However, in the MAASA framework the agent has access to the entire action distribution $\hat{a}_t$, rather than a single action, so it is more important to know how close the true and predicted action distributions are, rather than the accuracy. To quantify the distance between two distributions, the Wasserstein-1 distance [57] is used. Given two probability distributions μ, ν, the Wasserstein-1 distance is defined as

$$l_1(\mu, \nu) = \inf_{\gamma \in \Gamma(\mu,\nu)} \int_{M \times M} |x - y| d\gamma(x, y), \qquad (10)$$

where $\Gamma(\mu, \nu)$ is the set of probability distributions on $M \times M$, whose marginals are μ and ν. Given that for each state in the environment, there is an associated action action distribution, we record the mean and standard deviation of the Wasserstein distance between the predicted and true action distribution. In addition, to provide a baseline Wassterstein distance, we also compute the Wasserstein distance between a uniform distribution and the true action distribution. This provides insight into the worst case Wasserstein distance where the predicted action distribution is always uniform, or provides no useful information. The results of this analysis are shown in Table IX. It can be seen that in comparison to the baseline, the Wasserstein distance for the intention learning model is very small, providing a predicted action distribution that is representative of the true action distribution. This allows the ownship to have insight of the intruder's behavior and select actions accordingly, removing the assumption that agent's objectives need to be known a-priori.

TABLE VIII: Accuracy of the intention learning model on each case study.

Case Study	Accuracy
A	92%
B	92%
C	91%
D	89%

TABLE IX: Mean Wasserstein-1 distance for each case study.

Case Study	Intention Learning Model	Baseline
A	**0.03 ± 0.06**	0.41 ± 0.07
B	**0.02 ± 0.06**	0.42 ± 0.06
C	**0.03 ± 0.07**	0.42 ± 0.06
D	**0.03 ± 0.07**	0.42 ± 0.06

V. Conclusion

In this article, a decentralized autonomous separation assurance framework is proposed for handling heterogeneous high-density airspace sectors. With the intention learning model, agents are able to predict the intent of each intruder aircraft, reducing the non-stationarity in the environment, as well as allowing the agents to select actions that cooperate with the intruder's intent. By training the intention learning model with shared layers of the entire MAASA neural network, this reduces the overall number of training parameters and provides a more efficient training solution since batches of data can be jointly trained, rather than having to train one model before the other. In addition, given the decentralized implementation, the MAASA framework is able to computationally scale to any number of agents and is not restricted to a centralized controller. The results show that the MAASA framework significantly improves the state-of-the-art in heterogeneous airspace where agents are optimizing different reward functions. It is also shown through experiments that the intention learning model effectively learns the policy of the intruder aircraft in a limited number of training episodes. While this article focuses on en route structured airspace, the general approach of the MAASA framework allows it to be easily extended to 3D airspace environments or different reward formulations. Given these facts, this demonstrates the effectiveness of the article.

Acknowledgment

This research is partially funded by the National Science Foundation under Award No. 1718420, NASA Iowa Space Grant under Award No. NNX16AL88H, and the NVIDIA GPU Grant program.

References

[1] N. R. Council, *Autonomy research for civil aviation: toward a new era of flight*. National Academies Press, 2014.

[2] E. R. Mueller, P. H. Kopardekar, and K. H. Goodrich, "Enabling airspace integration for high-density on-demand mobility operations," in *17th AIAA Aviation Technology, Integration, and Operations Conference*, 2017, p. 3086.

[3] S. J. Undertaking, "U-space blueprint," *SESAR Joint Undertaking. Accessed September*, vol. 18, 2017.

[4] P. Kopardekar, J. Rios, T. Prevot, M. Johnson, J. Jung, and J. E. Robinson, "Unmanned aircraft system traffic management (utm) concept of operations," 2016. [Online]. Available: https://ntrs.nasa.gov/search.jsp?R=20190000370

[5] G. Hunter and P. Wei, "Service-oriented separation assurance for small uas traffic management," in *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*. IEEE, 2019, pp. 1–11.

[6] H. Erzberger and E. Itoh, "Design principles and algorithms for air traffic arrival scheduling," 2014. [Online]. Available: https://ntrs.nasa.gov/citations/20140010277

[7] B. T. Baxley, W. C. Johnson, J. Scardina, and R. F. Shay, "Air traffic management technology demonstration-1 concept of operations (atd-1 conops), version 3.0," 2016.

[8] E. Frazzoli, Z.-H. Mao, J.-H. Oh, and E. Feron, "Resolution of conflicts involving many aircraft via semidefinite programming," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 1, pp. 79–86, 2001.

[9] A. U. Raghunathan, V. Gopal, D. Subramanian, L. T. Biegler, and T. Samad, "Dynamic optimization strategies for three-dimensional conflict resolution of multiple aircraft," *Journal of guidance, control, and dynamics*, vol. 27, no. 4, pp. 586–594, 2004.

[10] P. J. Enright and B. A. Conway, "Discrete approximations to optimal trajectories using direct transcription and nmiscar programming," *Journal of Guidance, Control, and Dynamics*, vol. 15, no. 4, pp. 994–1002, 1992.

[11] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 1917–1922.

[12] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, "Model predictive control of swarms of spacecraft using sequential convex programming," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 6, pp. 1725–1740, 2014.

[13] D. Delahaye, C. Peyronne, M. Mongeau, and S. Puechmorel, "Aircraft conflict resolution by genetic algorithm and b-spline approximation," in *EIWAC 2010, 2nd ENRI International Workshop on ATM/CNS*, 2010, pp. 71–78.

[14] M. Pontani and B. A. Conway, "Particle swarm optimization applied to space trajectories," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 5, pp. 1429–1441, 2010.

[15] J. P. Chryssanthacopoulos and M. J. Kochenderfer, "Accounting for state uncertainty in collision avoidance," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 4, pp. 951–960, 2011.

[16] M. J. Kochenderfer, J. E. Holland, and J. P. Chryssanthacopoulos, "Next-generation airborne collision avoidance system," Massachusetts Institute of Technology-Lincoln Laboratory Lexington United States, Tech. Rep., 2012.

[17] J. P. Chryssanthacopoulos and M. J. Kochenderfer, "Decomposition methods for optimized collision avoidance with multiple threats," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 2, pp. 398–405, 2012.

[18] ——, "Hazard alerting based on probabilistic models," *Journal of guidance, control, and dynamics*, vol. 35, no. 2, pp. 442–450, 2012.

[19] H. Y. Ong and M. J. Kochenderfer, "Markov decision process-based distributed conflict resolution for drone air traffic management," *Journal of Guidance, Control, and Dynamics*, pp. 69–80, 2016.

[20] Y. Fu, X. Yu, and Y. Zhang, "Sense and collision avoidance of unmanned aerial vehicles using markov decision process and flatness approach," in *2015 IEEE International Conference on Information and Automation*, 2015, pp. 714–719.

[21] X. Yang and P. Wei, "Scalable multi-agent computational guidance with separation assurance for autonomous urban air mobility," *Journal of Guidance, Control, and Dynamics*, 2020.

[22] J. Bertram and P. Wei, "Distributed computational guidance for high-density urban air mobility with cooperative and non-cooperative collision avoidance," in *AIAA Scitech 2020 Forum*, 2020, p. 1371.

[23] X. Yang and P. Wei, "Autonomous free flight operations in urban air mobility with computational guidance and collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2021.

[24] M. J. Kochenderfer and J. Chryssanthacopoulos, "Robust airborne collision avoidance through dynamic programming," *Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-371*, 2011.

[25] K. D. Julian, M. J. Kochenderfer, and M. P. Owen, "Deep neural network compression for aircraft collision avoidance systems," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 3, pp. 598–608, 2018.

[26] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van Hasselt, D. Silver, T. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, and R. Tsing, "Starcraft ii: A new challenge for reinforcement learning," *arXiv preprint arXiv:1708.04782*, 2017.

[27] C. Amato and G. Shani, "High-level reinforcement learning in strategy games," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 75–82.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[29] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[30] M. Brittain and P. Wei, "Autonomous aircraft sequencing and separation with hierarchical deep reinforcement learning," in *Proceedings of the International Conference for Research in Air Transportation*, 2018.

[31] D.-T. Pham, N. P. Tran, S. Alam, V. Duong, and D. Delahaye, "A machine learning approach for conflict resolution in dense traffic scenarios with uncertainties," in *13th USA/Europe ATM R&D Seminar*, 2019.

[32] P. N. Tran, D.-T. Pham, S. K. Goh, S. Alam, and V. Duong, "An interactive conflict solver for learning air traffic conflict resolutions,"

Journal of Aerospace Information Systems, vol. 17, no. 6, pp. 271–277, 2020.
[33] M. Ribeiro, J. Ellerbroek, and J. Hoekstra, "Improvement of conflict detection and resolution at high densities through reinforcement learning," in *Proceedings of the International Conference for Research in Air Transportation*, 2020.
[34] R. Konda, H. M. La, and J. Zhang, "Decentralized function approximated q-learning in multi-robot systems for predator avoidance," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6342–6349, 2020.
[35] L. Zhang, Y. Sun, A. Barth, and O. Ma, "Decentralized control of multi-robot system in cooperative object transportation using deep reinforcement learning," *IEEE Access*, vol. 8, pp. 184 109–184 119, 2020.
[36] X. Chen, B. Fu, Y. He, and M. Wu, "Timesharing-tracking framework for decentralized reinforcement learning in fully cooperative multi-agent system," *IEEE/CAA Journal of Automatica Sinica*, vol. 1, no. 2, pp. 127–133, 2014.
[37] L. Li and W. Sheng, "Collision avoidance dynamic window approach in multi-agent system," in *2020 Chinese Automation Congress (CAC)*, 2020, pp. 2307–2311.
[38] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.
[39] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 285–292.
[40] R. E. Wang, M. Everett, and J. P. How, "R-maddpg for partially observable environments and limited communication," *ICML Workshop: Reinforcement Learning for Real Life*, 2019.
[41] I. Draganjac, D. Miklić, Z. Kovačić, G. Vasiljević, and S. Bogdan, "Decentralized control of multi-agv systems in autonomous warehousing applications," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 4, pp. 1433–1447, 2016.
[42] W. Dong, S. Liu, Y. Ding, X. Sheng, and X. Zhu, "An artificially weighted spanning tree coverage algorithm for decentralized flying robots," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1689–1698, 2020.
[43] P. K. Menon, G. D. Sweriduk, and B. Sridhar, "Optimal strategies for free-flight air traffic conflict resolution," *Journal of Guidance, Control, and Dynamics*, vol. 22, no. 2, pp. 202–211, 1999.
[44] S. Wollkind, J. Valasek, and T. Ioerger, "Automated conflict resolution for air traffic management using cooperative multiagent negotiation," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004, p. 4992.
[45] M. Brittain and P. Wei, "Autonomous separation assurance in an high-density en route sector: A deep multi-agent reinforcement learning approach," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3256–3262.
[46] ——, "One to any: Distributed conflict resolution with deep multi-agent reinforcement learning and long short-term memory," in *AIAA Scitech 2021 Conference*, 2021.
[47] H. Niu, C. Ma, P. Han, and J. Lv, "An airborne approach for conflict detection and resolution applied to civil aviation aircraft based on orca," in *2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, 2019, pp. 686–690.
[48] P. Zhao and Y. Liu, "Physics informed deep reinforcement learning for aircraft conflict resolution," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2021.
[49] J. Mollinga and H. van Hoof, "An autonomous free airspace en-route controller using deep reinforcement learning techniques," *arXiv preprint arXiv:2007.01599*, 2020.
[50] M. Brittain, X. Yang, and P. Wei, "A deep multi-agent reinforcement learning approach to autonomous separation assurance," *arXiv preprint arXiv:2003.08353*, 2020.
[51] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
[52] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.
[53] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*, 2017, pp. 6379–6390.
[54] 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, 2016, pp. 1–8.
[55] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, and I. Stoica, "Ray: A distributed framework for emerging ai applications," in *13th USENIX Symposium on Operating Systems Design and Implementation OSDI 2018*, 2018, pp. 561–577.
[56] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, "Dealing with non-stationarity in multi-agent deep reinforcement learning," *arXiv preprint arXiv:1906.04737*, 2019.
[57] C. Villani, *Optimal transport: old and new*. Springer Science & Business Media, 2008, vol. 338.

Marc Brittain is a fourth-year Ph.D. candidate in the Department of Aerospace Engineering at Iowa State University. He is currently working as a research assistant under Dr. Peng Wei who leads the Intelligent Aerospace Systems Laboratory (IASL). He received his bachelor's of science degree in Physics with a minor in mathematics from the University of North Carolina at Wilmington. His research interests include Deep Reinforcement Learning, Multi-Agent Reinforcement Learning, Machine Learning, Air Traffic Management with applications in UAS Traffic Management (UTM) and Air Traffic Control (ATC).

Peng Wei is an assistant professor in the Department of Mechanical and Aerospace Engineering at the George Washington University. By contributing to the intersection of control, optimization, machine learning, and artificial intelligence, he develops autonomy and decision support tools for aeronautics, aviation and aerial robotics. His current focus is on safety, efficiency, and scalability of decision making systems in complex, uncertain and dynamic environments. His recent applications include: Air Traffic Control/Management (ATC/M), Airline Operations, UAS Traffic Management (UTM), eVTOL Urban Air Mobility (UAM) and Autonomous Drone Racing (ADR). He is leading the Intelligent Aerospace Systems Lab (IASL). He received his Ph.D. degree in Aerospace Engineering from Purdue University in 2013.