Real-time Deep Reinforcement Learning for Evacuation under Emergencies

January 2024

Yujing Zhou, Yupeng Yang, Dahai Liu, Yongxin Liu, Sirish Namilae, Houbing Song

Embry-Riddle Aeronautical University

US DEPARTMENT OF TRANSPORTATION GRANT

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the Department of Transportation, University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

4. Title and Subtitle Real-time Deep Reinforcement Learning for Evacuation under Emergencies	5. Report Date: 01/31/2024
	6. Source Organization Code : ERAU Cost Center 61687
7. Author(s) Yujing Zhou, Yupeng Yang, Dahai Liu, Yongxin Liu, Sirish Namilae, Houbing Song	8. Source Organization Report No. CATM-2024-R2.ERAU
9. Performing Organization Name and Address Center for Advanced Transportation Mobility Transportation Institute 1601 E. Market Street Greensboro, NC 27411	10. Work Unit No.
	11. Contract or Grant No. 270218CC
12. Sponsoring Agency Name and Address University Transportation Centers Program (RDT-30) Office of the Secretary of Transportation–Research U.S. Department of Transportation 1200 New Jersey Avenue, SE Washington, DC 20590-0001	13. Type of Report and Period Covered Final Report: 10/01/2021-12/31/2023
	14. Sponsoring Agency Code: USDOT/OST-R/CATM
15. Supplementary Notes:	

16. Abstract
This study adopted A3C algorithm to simulate the evacuation process under different situations (multi agents and difference environment conditions), and results were compared with Deep Q-Networks (DQN), to demonstrate the efficiency and effectiveness of A3C algorithm using in evacuation models. Results indicated that under static environments, A3C demonstrated superior adaptability and quicker response times. Furthermore, with an increasing number of agents, A3C showed better scalability and robustness in managing complex interactions and providing quick evacuations. These outcomes highlight A3C's advantage over traditional RL models under varying and challenging conditions. The report concludes with a discussion of the practical implications and benefits of these models. It emphasized their potential in enhancing real-world evacuation planning and safety protocols.

18. Key Words Reinforcement Learning, Multi-agent collaboration, emergency, airport evacuation	19. Distribution Statement Unrestricted; Document is available to the public through the National Technical Information Service; Springfield, VT.

20. Security Classif. (of this report) unclassified	21. Security Classif. (of this page) unclassified	22. No. of Pages 80	23. Price …

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Executive Summary

An emergency could happen at anytime and anywhere in daily life. It is a broad term that summarizes several different situations, which are generally serious, unexpected, and require immediate action. This study focused on aviation emergency evacuation situations, which has the characteristic of high level of time pressure and uncertainty. Therefore, a real-time decision-making assistance system is needed and will be largely helpful when an emergency exists at the airport or on an aircraft when evacuation occurs. The system and model built in this paper are based on the Asynchronous Advantage Actor Critic (A3C) algorithm, which is one of the newest Deep Reinforcement Learning Algorithms. The A3C algorithm provides quicker and more efficient evacuation routes for the agents in the environment compared to traditional evacuation simulation models, thus saving time for passengers. This study adopted the A3C algorithm as a tool to simulate the evacuation process under different situations (multi agents and difference environment conditions), and results were compared with Deep Q-Networks (DQN), to demonstrate the efficiency and effectiveness of using the A3C algorithm in evacuation models. Results indicated that under static environments, A3C demonstrated superior adaptability and quicker response times. A3C performed 43.86% faster than DQN in terms of the average time taken for agent evacuation and converged quicker at around 100 episodes comparing to 250 episodes for DQN algorithm. In scenarios with moving threats, the A3C algorithm also outperformed DQN in terms of dynamic pathfinding efficiency and maintaining agent safety. Furthermore, with an increasing number of agents, A3C demonstrated better scalability and robustness in managing complex

interactions and providing quick evacuations for multiple agents. These outcomes highlight A3C's advantage over traditional models, especially in terms of adaptability, efficiency, and scalability under varying and challenging conditions. This report concludes with a discussion of the practical implications and benefits of these models, emphasizing their potential in enhancing real-world evacuation planning and safety protocols. The practical implications and benefits of these models are also discussed at the end of this report.

Table of Contents

Abstract

An emergency could happen at anytime and anywhere in daily life. It is a broad term that summarizes several different situations, which are generally serious, unexpected, and require immediate action. This study focused on aviation emergency evacuation situations, which has the characteristic of high level of time pressure and uncertainty. Therefore, a real-time decision-making assistance system is needed and will be largely helpful when an emergency exists at the airport or on an aircraft when evacuation occurs. The system and model built in this paper are based on the Asynchronous Advantage Actor Critic (A3C) algorithm, which is one of the newest Deep Reinforcement Learning Algorithms. The A3C algorithm provides quicker and more efficient evacuation routes for the agents in the environment compared to traditional evacuation simulation models, thus saving time for passengers. This study adopted the A3C algorithm as a tool to simulate the evacuation process under different situations (multi agents and difference environment conditions), and results were compared with Deep Q-Networks (DQN), to demonstrate the efficiency and effectiveness of using the A3C algorithm in evacuation models. Results indicated that under static environments, A3C demonstrated superior adaptability and quicker response times. A3C performed 43.86% faster than DQN in terms of the average time taken for agent evacuation and converged quicker at around 100 episodes comparing to 250 episodes for DQN algorithm. In scenarios with moving threats, the A3C algorithm also outperformed DQN in terms of dynamic pathfinding efficiency and maintaining agent safety. Furthermore, with an increasing number of agents, A3C demonstrated better scalability and robustness in managing complex

interactions and providing quick evacuations for multiple agents. These outcomes highlight A3C's advantage over traditional models, especially in terms of adaptability, efficiency, and scalability under varying and challenging conditions. This report concludes with a discussion of the practical implications and benefits of these models, emphasizing their potential in enhancing real-world evacuation planning and safety protocols. The practical implications and benefits of these models are also discussed at the end of this report.

Section 1: Introduction

Every second counts in the face of an emergency. Emergencies often threaten property, the environment, and, most importantly, human lives. Among various types of emergency situations, those occurring in transportation sectors such as aviation require immediate attention and care. Given the high volume of people constantly moving across various modes of transportation, any emergency during normal operations could cause major disruptions to schedules and put thousands of lives at risk. In the aviation industry, emergency management is a critical and highly regulated area focused on ensuring the most effective and efficient responses under intense time pressure and mental stress. Emergency situations demand swift, precise, and efficient responses to minimize damage and prevent future occurrences. Being both reactive and proactive is essential for effectively managing emergencies that occur in aircraft and airports. Therefore, a comprehensive decision-making system capable of identifying the most time-efficient routes for emergency evacuation at airports is essential.

Safety and compliance are of paramount importance in the aviation industry. According to the Federal Aviation Administration (FAA), over 45,000 flights are

managed daily, with approximately 2.9 million passengers flying in and out of the United States (FAA, 2022). Given such a large volume of daily flights, passenger safety is a top priority for airlines and airports. According to airline on-time statistics and delay causes (Figure 1) reported by the Bureau of Transportation Statistics (BTS) in 2021, the five broad categories most affecting normal airline and airport operations are: (1) Air carrier delays, (2) Extreme weather conditions, (3) The National Aviation System (NAS), (4) Late-arriving aircraft, and (5) Security concerns. Each of these factors can cause significant delays and elevate risk during operations.

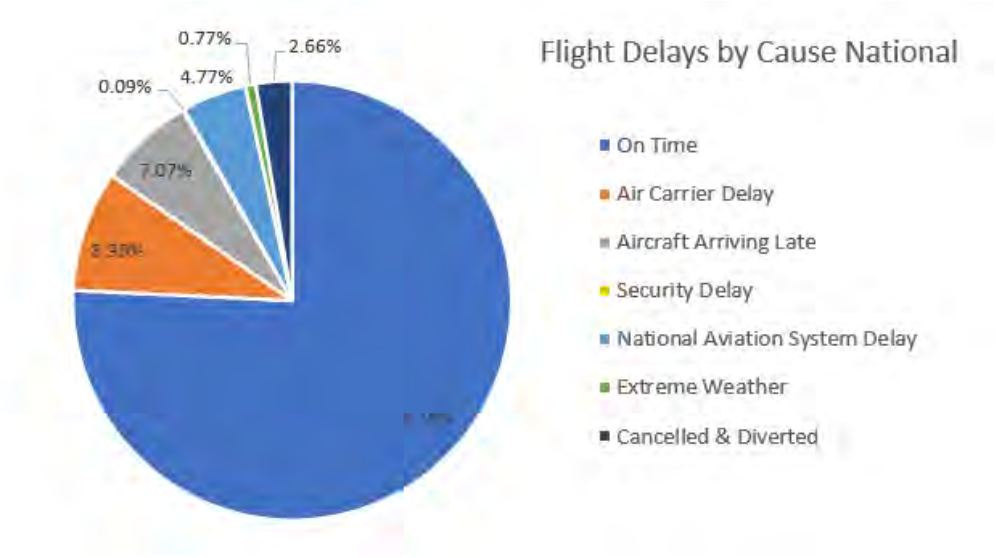

Fig. 1. Flight Delays by Cause National

The structure of this paper is as follows: Section 2 presents a comprehensive literature review of recent research focused on simulating and managing emergency situations across various environments. This section explores two primary types of emergencies in the aviation industry, assessing their impacts on both aircraft and airport operations. It also examines relevant regulations, evacuation plans, and prior studies on airport evacuation. Additionally, it analyzes the implementation of algorithms designed to

optimize evacuation routing and planning. The algorithms and models reviewed include, but are not limited to, the Asynchronous Advantage Actor-Critic (A3C) algorithm, Multi-Agent Deep Reinforcement Learning (MADRL), Velocity Obstacle (VO), and Social Force (SF) models. Section 3 outlines the methodology, model development, and system architecture, including theoretical frameworks and the design of simulation environments. Section 4 presents the results of the evacuation simulations and provides a detailed analysis of the findings. Finally, Section 5 offers a comprehensive conclusion of the study and discusses potential future improvements and research directions.

Section 2: Literature Review

Emergency Situations

An emergency is a broad term that encompasses various unexpected situations with the potential to cause serious consequences and that require immediate response. As defined by the Oxford English Dictionary, an emergency is 'a serious, unexpected, and often dangerous situation requiring immediate action.' This definition highlights the key characteristics of an emergency: unpredictability, potential danger, and the urgent need for prompt action (Alexander, 2013). An emergency can easily disrupt operations due to its unpredictable nature and may lead to severe consequences if not managed promptly and effectively. Therefore, mitigations from both proactive and reactive aspects are important to protect lives and properties from emergency situations. Depending on the type of emergency, the time required for information processing and response can vary significantly; consequently, mitigation and evacuation plans must be adapted accordingly.

Examples of emergency situations include floods, hurricanes, fire hazards, traffic accidents, blizzards, hail, etc. Each situation has a different scale and severity, which will have varying consequences and damage. While many emergency situations can be partially mitigated by adhering to standard operating procedures, reactive measures, such as evacuation plans and real-time decision-making systems, are crucial for effectively and accurately responding to emergencies. For example, when a fire suddenly breaks out in a movie theater or an aircraft makes an emergency landing at an airport, the situation can often be effectively managed by following established emergency response procedures and evacuation plans, minimizing disruption to normal operations However, there are emergency situations that cannot be spotted and reacted to immediately, such as natural disasters and terrorist attacks. When large-scale emergency situations happen, the normal emergency response plan will not be fast enough to respond to protect lives and properties (Alexander, 2013). Therefore, real-time decision-making systems and swift human responses are essential and must be activated immediately to help ensure better outcomes during emergency situations.

Conducting risk assessments is an effective way to better understand emergency situations, enabling more efficient and safer planning and response. To evaluate the safety of evacuation when a large steel gymnasium collapses due to a localized fire, Zhang et al. (2016) proposed a steel temperature rise model that accounts for both smoke thermal radiation and the thermal radiation from fire acting on steel components. Since the physical collapse of a steel gymnasium structure has a more significant impact on evacuation than the smoke hazard, a method was developed to quantitatively assess the casualties caused by the collapsed structure. Additionally, a quantitative risk assessment

of evacuation safety was performed comparing the Available Safe Egress Time (ASET) and Required Safe Egress Time (RSET). The result of the experiment showed that the modified temperature rise model of steel components accurately predicted the temperature rise in the fire. By considering factors including, the distance from the furthest point to the evacuation exit, different moving speeds, the width of the evacuation exit, and the density of people, the experiments demonstrated that the model could accurately predict the evacuation time in the scenario.

Aviation Emergency Situations

Emergency situations in aviation can be classified into two categories: aircraft emergencies and airport emergencies. Aircraft emergencies can be complex, dangerous, and may be caused by several different points of failure. Aircraft emergencies, including instrument failure, autopilot failure, landing gear malfunction, engine failure, etc., can happen at any time while in flight. To deal with these unpredictable emergency situations, the FAA and manufacturers have published aircraft model-specific emergency response plans and checklists to help pilots deal with emergencies and make informed decisions under time pressure. For example, in the event of an in-flight engine failure, pilots typically have three emergency landing options: precautionary landing, forced landing, and ditching. A precautionary landing is considered the safest emergency landing option, with a fatality rate of only 0.06%, making it the most likely to result in survival. However, forced landings and ditching place significant pressure on pilots and often provide limited time for reaction, resulting in higher fatality rates of up to 10% and 20%, respectively. (Rossier, n.d.). The fatality rates not only highlight the dangers

associated with engine failure but also emphasize the critical importance of precautionary measures and proactive planning.

Compared with aircraft emergencies, airport emergency situations are less frequent but poses an equal or even more significant threat to lives and properties. According to Advisory Circular (AC) 150/5200-31C published by the U.S. Department of Transportation (DOT) and FAA, an airport emergency is “any occasion or instance, natural or man-made, that warrants action to save lives and protects property and public health.” (DOT & FAA, 2009). More specifically, the FAA and DOT classify various types of airport emergencies, including but not limited to: (Villamizar, 2022):

- airport and aircraft malfunctions that impede safety of flight.
- suspicious packages.
- bomb threats.
- sabotage of aviation-related equipment.
- structural fires and non-structural fires.
- natural disasters; etc.

To deal with the different types and severities of airport emergency situations, the FAA and DOT require each certificate holder of an airport to have the ability to introduce and maintain an airport emergency plan (AEP). The AEP must have responses plans handle different emergencies that may arise and maximize the ability to protect people and property (DOT & FAA, 2004). A certified AEP typically follows four key phases, mitigation, preparedness, response, and recovery, as recommended by the Federal Emergency Management Agency (FEMA) for effective emergency management planning.

Various factors can trigger emergency situations involving both aircraft and airports, most of which require immediate evacuation and other prompt measures to mitigate potential consequences. For example, in 2016, the right engine of an American Airlines Boeing 767 caught on fire during takeoff. Although an emergency evacuation was successfully carried out at Chicago O'Hare International Airport, 20 individuals sustained varying degrees of injuries, while all 161 passengers were evacuated using the emergency slides (Hradecky, 2016). Another example is an evacuation that happened at Denver International Airport caused by a left engine fire on a United Airlines Bombardier CRJ-700 in 2017. In this evacuation, 59 passengers plus four crewmembers were safety evacuated, with no injuries reported (Hradecky, 2017). Also in 2017, an Airbus A320 performed an emergency landing at Daytona Beach International Airport due to hail damage cracking the front windshield. The 132 passengers and crewmembers were safely evacuated because of the efficient evacuation plan and design that was followed (Hradecky, 2017). In all three emergencies, accurate execution of the evacuation plan played an important role for safety. In 2017, the FAA established a regulation known as the '90-second evacuation rule,' requiring that all passengers be able to evacuate an aircraft within 90 seconds using only half of the available emergency exits/slides. This rule standardized the evacuation timeline across different aircraft types and designs, promoting efficient and timely evacuations. However, given the complexity of emergency scenarios, the aviation industry must adopt additional measures to further mitigate their impact. Accordingly, evacuation procedures and emergency response systems can be categorized into two main areas: aircraft evacuation and airport evacuation, which will be discussed in more detail in the following sections.

Airport Evacuation Simulation

To achieve the goal of safety and efficiency, evacuation plans and routes play important roles for airlines and airports. Unlike aircraft evacuation, airport evacuation does not have a rule or regulation that sets the maximum time frame for evacuation. In this case, time is not a variable nor a criterion to evaluate the effectiveness of an evacuation. One of the most significant difficulties in airport evacuation and AEP is that it is impossible to apply a generic plan to all different designs and layouts of airports. To design and evaluate an effective airport evacuation, researchers need to explore and build the model based on the unique characteristics of different airports due to different layouts and designs, which is a time-consuming and complex task.

Chen et al. (2019) conducted an agent-based simulation for airport evacuation to investigate the optimal number of exit doors and the best path for passengers to evacuate when an emergency occurs. In the study, the researchers mainly focused on how to improve the efficiency of evacuation under emergency situations at the airport and used a local airport as their base for the model. The results also evaluated if the evacuation strategies currently used are adequate with the continuously growing number of passengers. The agent-based simulation was modeled by using the AnyLogic simulation software, which can simulate the routes and evacuation process for airports while changing the variables, including exit doors, evacuating path, etc. (Chen et al., 2019).

The results from simulations involving varying input variables demonstrated a clear relationship between passenger volume and total evacuation time. As expected, higher passenger volumes led to longer evacuation times. Additionally, the number and placement of exit doors were identified as two critical factors significantly influencing

evacuation efficiency. In the simulation, passengers tended to evacuate through exits that were closer and more accessible, which potentially increased the time of evacuation because of congestion around those exits. Thus, Chen et al. (2019) recommended that each airport authority develop a customized exit plan that accounts for evacuation paths, as well as the number and placement of emergency exits within the airport. The simulation and methodology can be used to estimate evacuation times for different airports based on their specific construction and design. While the simulation is valuable for optimizing evacuation plans and routes, it can be further improved by accounting for agents, such as elderly individuals and children, who move at slower speeds. Additionally, incorporating the location of the emergency or threat as a variable could enhance evacuation efficiency by allowing for adaptive routing and the development of multiple contingency plans.

The most important consideration in airport operations is safety. Because full-scale evacuation drills are costly and time-consuming, computer modeling techniques are widely adopted for simulating evacuations. Cheng et al. (2014) developed an agent-based model to simulate the passenger evacuation process. In the model, the airport was treated as a complex system divided into two main areas: landside and airside. Airport patrons were categorized into two groups, travelers (passengers) and non-travelers. Three emergency exits were designated on each side of the airport. Additionally, passengers were assigned to one of three security levels based on their location, which determined the specific exit routes they followed during evacuation. As shown in Figure 2, passengers activities were sorted into processing activities that were mandatory for travelers aboard the airplane and discretionary activities, which were defined as any other

activities conducted during the non-processing time. Processing activities include check-in, security, customs, and boarding, whereas discretionary activities include walking, shopping, eating, etc. Two levels of behavioral responses were considered in the model. The global behavioral level defined the overall evacuation process, which included responding to the emergency signal, moving toward exits, waiting near exits, and completing the evacuation. At the local behavioral level, the model accounted for variations in passenger response times based on factors such as age and travel purpose. It also assumed that passengers traveling in groups would adjust their pace to match the slowest member. Additionally, the model assumed that no panic behavior occurred during the evacuation.

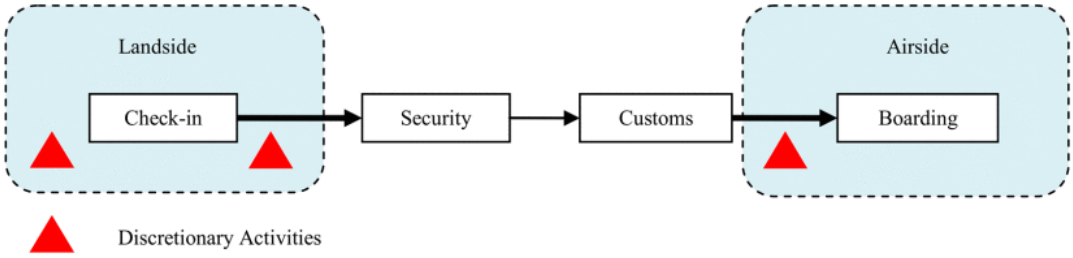

Fig. 2. The Departure Process of Passengers in Airport (Cheng et al. 2014)

The simulation results demonstrated that passengers traveling in groups required more time to evacuate compared to individual travelers, as group dynamics involved additional time for decision-making, movement coordination, and waiting for other members. The experiment also confirmed that the agent-based model is effective for analyzing pedestrian group behavior in complex environments. However, the model was not designed to simulate extreme emergency scenarios, and the role of airport staff was not incorporated into the simulations.

Individuals with disabilities are often the most affected during emergencies due to varying physical and mental conditions, and the built environment can present additional challenges not typically faced by the general population. However, previous studies have largely overlooked the inclusion of persons with disabilities in evacuation modeling and simulated environment design. Recognizing the importance of incorporating disability considerations into emergency evacuation planning and preparedness, Christensen and Sasaki (2008) developed the BUMMPEE model, an agent-based simulation framework. BUMMPEE, which stands for Bottom-Up Modeling of Mass Pedestrian Flows—Implications for the Effective Egress of Individuals with Disabilities, classified built environments based on their physical characteristics and simulated diverse populations using individual variable criteria. This model enables a more accurate representation of the behaviors and needs of individuals with disabilities, offering insights into how effectively a built environment supports their safe evacuation during emergencies.

To develop their model, Christensen and Sasaki (2008) evaluated the impact of both existing and proposed Americans with Disabilities Act Accessibility Guidelines (ADAAG) on the built environment. They identified five distinct disability groups and six criteria that vary depending on the type of disability, as well as four environmental characteristics that significantly affect individuals with disabilities. The six disability groups included physical disability, mental disability, go-outside-home disability, sensory disability, and self-care disability. Moreover, the six criteria were specified as individual speed, individual size, individual ability to negotiate the terrain, individual perception, individual psychological profile, and individual assistance, whereas the BUMMPEE model addressed these criteria by including seven different populations: motorized

wheelchair users, non-motorized wheelchair users, the visually impaired, the hearing impaired, the stamina impaired, individuals without disabilities familiar with the environment, and individuals without a physical or sensory disability but less familiar with the environment. Each population was separately defined by the difference in speed, size, and ability to negotiate the terrain. The four environmental characteristics were classified as exit character, route character, obstacle character, and planned systems.

After setting the basic criteria, population, and environmental characteristics, the BUMMPEE model was written in C++ with the use of a common graphical interface structure. In order to validate the reasonableness of the model, a physical evacuation of the same environment and population was conducted at Utah State University Human Services Research Center (HSRC) on September 14, 2005, to compare model results with actual results. However, due to differences in the data available for the physical and simulated models, only two measurements (Maximum evacuation time and the number of evacuees evacuating at each exit) were used for comparison. The results validated the consistency and similarity of the BUMMPEE model with real-world scenarios, indicating that the average total evacuation time of the evacuation simulations was 33 seconds less than the physical evacuation.

Application of Machine Learning in Evacuation

Machine learning, a study “of computer algorithms that can improve automatically through experience and data” (Mitchell, 1997), has been continuously developed and implemented by researchers in several different areas, including medical, image recognition, computer vision, product recommendation, etc. In the aviation

industry, machine learning serves as a powerful tool for enhancing the safety and efficiency of the thousands of flights operating each day.

Compared to private vehicle travel, public transportation systems, such as metros, buses, and airports, are inherently more complex and present a higher likelihood of emergencies. Due to unpredictable variables, including constantly changing passenger volumes, emergencies in public transportation systems often have more severe consequences and require immediate mitigation or evacuation. To address this challenge, Ma et al. (2022) applied machine learning algorithms to forecast passenger flow under emergency conditions in metro systems. They emphasized that due to significant passenger congestion during peak hours, it is essential to develop a reliable passenger flow forecasting system. To achieve this, they implemented a Long Short-Term Memory (LSTM) model, a type of machine learning algorithm well-suited for time-series prediction.

In their study, Ma et al. (2022) employed transfer learning in combination with a Long Short-Term Memory (LSTM) network to predict passenger flow in metro systems under both normal and emergency conditions. They noted that while Gao et al. (2019) had previously analyzed changes in passenger flow using traditional mathematical modeling methods, these approaches struggle to effectively handle the vast volume of passenger movement during emergencies. Therefore, the integration of LSTM networks

with transfer learning offers a more suitable and scalable solution for forecasting large-scale passenger flow in emergency scenarios compared to conventional methods.

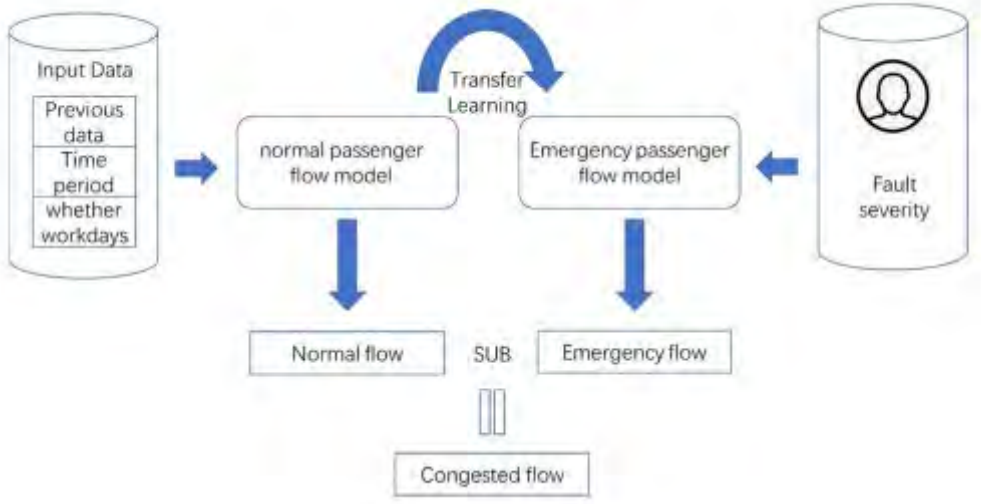

Fig. 3. The Transfer Learning Procedure Between Normal and Emergency Passenger Flow (Ma et al., 2022)

Ma et al. (2022) structured the model into two components: a normal passenger flow model and an emergency passenger flow model. They first developed the normal passenger flow model and then applied transfer learning to adapt it into the emergency passenger flow model, enabling the prediction and simulation of passenger flow during emergency scenarios (Figure 3). The results indicated that machine learning algorithms are highly effective in assessing the capacity and dynamics of public transportation systems such as airports and metro networks. The research further demonstrated that these algorithms can sensitively detect fluctuations in passenger flow and are capable of handling both high and low volumes under emergency conditions.

Gota et al. (2020) approached airport emergency management from a proactive standpoint by leveraging machine learning algorithms for threat object detection. In their

research, they implemented a convolutional neural network (CNN) and specialized libraries to detect threats, such as explosives, using X-ray images. This application represents the "proactive" level of an airport emergency plan, aiming to identify threats early and minimize evacuation time. The results demonstrated that machine learning significantly enhances threat detection capabilities, thereby improving evacuation efficiency and contributing to the protection of lives and property.

Similarly, Zarraonandia et al. (2009) developed a virtual environment using DimensioneX to simulate various airport emergency scenarios. Their system allowed researchers to identify and analyze different emergency causes and their corresponding consequences. The results indicated that the virtual environment could serve as an effective evaluation and training tool, enabling supervisors and researchers to design more effective airport emergency plans. This approach provides a valuable platform for testing emergency decision-making systems and multi-agent collaborative evacuation models, allowing for continuous improvement of emergency response strategies.

In the field of emergency search and rescue, robotic systems can significantly reduce human risk, minimize property loss, and enhance the speed of locating and rescuing survivors. With the goal of developing a fully automated system, Vaidyanath et al. (2020) created a simulation framework combining a swarm of unmanned aerial vehicles (UAVs) and a virtual “spokesperson” in a virtual reality environment. Their system was designed to determine when to rely on autonomous agents, such as UAVs and spokespersons, and when to prompt human operator intervention.

The experiment simulated a wildfire threatening a small town, where UAVs and the spokesperson collaborated with a human operator to locate and guide survivors to

safety. The UAVs were equipped with pre-recorded warning messages and tasked with convincing civilians to evacuate before the fire reached them. The virtual spokesperson assisted with negotiation, while the human operator was prompted when the UAV was unable to persuade a group. The spokesperson was implemented using a Wizard of Oz (WoZ) setup to collect real-world interaction data between the system and human operators.

Vaidyanath et al. (2020) applied reinforcement learning to develop an adaptive strategy for the UAVs, enabling them to find survivors, encourage evacuation, and guide them to safety. The RL problem was modeled as a Markov Decision Process (MDP) and explored using both Monte Carlo simulation and Temporal Difference Learning. The on-policy Monte Carlo algorithm yielded the best performance. The system included three levels of communication with survivor groups: UAV warnings, spokesperson persuasion, and human operator intervention. Survivor groups were modeled as three types: a stubborn couple, an elderly couple, and a babysitter with a child.nSeven policy actions were defined: warn, allow-spokesperson-to-negotiate, interrupt-operator, query-for-guidance-info, UAV-guide, vehicle-guide, and wait. Additionally, five state variables were established: operator business level (0–3), group status (0–5), fire approach time (0–4), preferred guidance type (1–3), and negotiate status (0–2), resulting in 1,440 unique states and 10,080 state-action pairs. Three different fire approach times (2, 3, and 4) were used to train distinct policies for all survivor groups. Each policy was trained over 1 million episodes and tested across 10,000 episodes. Results indicated that the success rate of rescuing survivors exceeded 95% when the fire approach time was set to 4, averaged 90% at a setting of 3, and reached 74% at a setting of 2.

These studies highlight the diverse and growing applications of machine learning in emergency management and evacuation scenarios. Among them, reinforcement learning stands out as a particularly powerful approach due to its ability to operate effectively in uncertain and dynamic environments. The next section presents a focused review of reinforcement learning and its relevance to emergency evacuation systems.

Reinforcement Learning (RL)

As one of the most promising and effective machine learning approaches, reinforcement learning is distinguished by its ability to function without requiring a supervisor or a complete model of the environment (Sutton & Barto, 2018). Leveraging this characteristic, a reinforcement learning agent autonomously learns to make decisions by interacting with its environment, aiming to achieve a defined goal while maximizing cumulative rewards. Through repeated trial and error, the agent refines its strategies, progressing from basic decision-making to complex, adaptive behaviors that optimize outcomes. In emergency scenarios, one of the most challenging tasks is evacuating crowds both safely and efficiently. Poorly chosen evacuation routes or strategies can significantly increase casualties. Additionally, the unpredictability of crowd movement further complicates evacuation planning. However, by modeling crowd behavior and movement trajectories, researchers can simulate likely outcomes and develop more effective evacuation strategies.

To address limitations in traditional evacuation simulation models, Yao et al. (2019) introduced a reinforcement learning-based data-driven crowd evacuation (RL-DCE) framework. Their model first established a dynamic crowd evacuation (DCE) foundation, extracting position and velocity data from video footage to quantify crowd

cohesiveness. They developed a cohesiveness-based K-means (C-K-means) clustering algorithm to group individuals and predict their movement trajectories. The framework also introduced a hierarchical path planning mechanism composed of two layers: a top layer that applied RL algorithms to train control policies using the predicted trajectories, and a bottom layer that ensured collision avoidance using the Reciprocal Velocity Obstacles (RVO) model.

Yao et al. (2019) validated the effectiveness of group-based path computation by comparing it with individual-based approaches. The results indicated that the simulated trajectories closely matched real-world video data, demonstrating the model's accuracy. They further analyzed path control through parameter weight adjustments and found that the system effectively adapted to dynamic environments. After 180 iterations, the RL-based path planning algorithm converged, showing stable performance. Compared to other methods, RL-DCE improved the visual realism of crowd simulation while maintaining adaptability in changing scenarios.

Zhang and Guo (2014) approached the evacuation challenge by developing a distributed multi-robot system designed to guide people during emergencies. Their system incorporated a cooperative exit-seeking algorithm that allowed robots to estimate and follow gradient fields for optimal movement. The model also integrated two panic behavior models to better simulate human reactions. Experiments conducted in a simulated mall environment with 130 and 250 evacuees demonstrated that the system could reduce evacuation time by approximately 50% and 40%, respectively.

Focusing on fire evacuations in confined environments, Tian and Jiang (2018) conducted large-scale evacuation experiments in the world's largest immersed tube tunnel

test base. Recognizing the unique challenges posed by the enclosed and narrow design of tunnels, they studied the effects of high temperatures on evacuation speed and calculated the minimum safe escape time using a mathematical algorithm. Experiments conducted at the Hong Kong-Zhuhai-Macau experimental base revealed that temperatures exceeding 60°C, unbearable for breathing, could occur within 25 meters upstream and downstream from a 50MW fire source. By transforming the tunnel evacuation problem into an origin-destination (O-D) path selection model, the authors applied reinforcement learning to identify optimal escape paths. Results inidcaited that convergence in path usage as iteration counts increased.

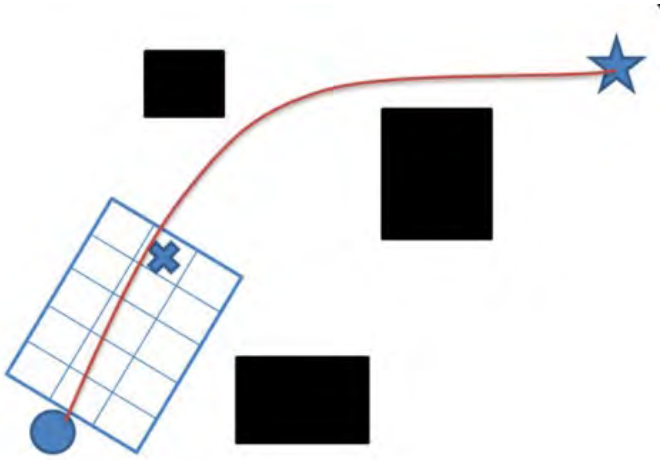

Fig. 4. Illustration of Robot Adaptive Path Planning Using Inverse Reinforcement Learning (Kim & Pineau, 2015)

Another notable application of RL in emergency planning was presented by Kim and Pineau (2015), who developed a framework for adaptive path planning in dynamic human environments (Figure 4). Their system integrated three modules: feature extraction, inverse reinforcement learning (IRL), and path planning. The IRL component learned from expert-generated demonstration trajectories to model optimal behavior in

environments filled with moving agents. The framework was tested in various scenarios, including uncontrolled dynamic settings, and showed strong adaptability, making it particularly applicable to airport emergencies, where passengers must navigate chaotic and unpredictable spaces to reach safety.

These studies collectively demonstrate the diverse potential of reinforcement learning in emergency evacuation systems. Whether applied to threat detection, crowd dynamics, robot-guided evacuations, or path planning in constrained spaces, RL provides a robust and adaptive framework for managing uncertainty and improving outcomes. Its ability to learn from experience and adapt to changing conditions makes it a critical tool for designing next-generation evacuation and emergency response systems.

Asynchronous Advantage Actor Critic (A3C)

Asynchronous Advantage Actor-Critic (A3C) is recognized as one of the most efficient algorithms in the field of reinforcement learning. In this framework, agents learn by repeatedly interacting with the environment, taking actions, receiving rewards, and updating value estimates, ultimately optimizing their policies to converge on the best possible outcomes. A key strength of A3C lies in its use of asynchronous training, which significantly enhances training speed and efficiency. Unlike traditional reinforcement learning algorithms that rely on synchronous updates and may suffer from bottlenecks or redundant data, A3C deploys multiple agents in parallel, each interacting with separate copies of the environment. These agents explore independently while contributing collectively to the learning process. This parallelized approach enables the collection of diverse experiences, reducing correlations in the training data and enhancing overall learning stability. A3C updates the global network parameters asynchronously, with

agents pushing updates immediately after completing their batch of experiences, without waiting for others to finish. This mechanism not only accelerates the learning process but also allows the global policy to benefit from the aggregated experiences of all agents in real time, promoting faster adaptation and more robust convergence.

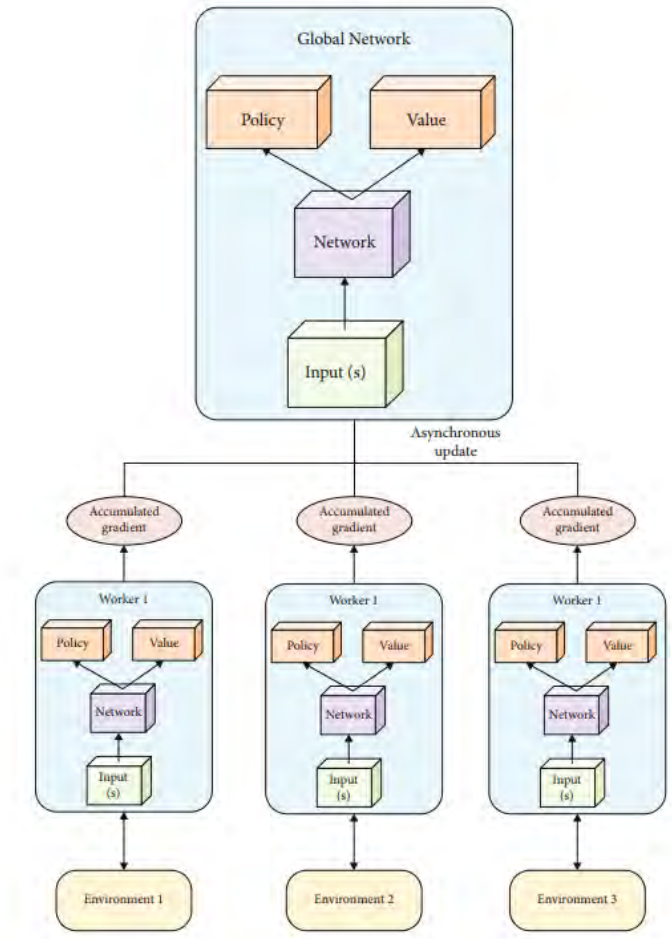

Fig. 5. The A3C Algorithm Framework (Lee & Yoo, 2023)

Research conducted by Ding et al. (2021) illustrated the advantages of using Asynchronous Advantage Actor Critic (A3C) in optimizing the routing path for data transmission evaluation. In their research, Ding et al. (2021) implemented the delay tolerant networks (DTN) and applied Deep Q-learning Network (DQN) and A3C under

different scenarios. The results show that A3C can get close to the top value with fewer episodes compared to the DQN algorithm, which indicates the efficiency and improvements of the A3C algorithm in node and link equilibrium. By using A3C, the evacuation model in a different situation could reach its optimal value and converge in a relatively short amount of time.

Effect of Human Psychology and Behavior During Evacuations

When emergencies occur, panic often sets in, and individuals exhibit a wide range of behaviors and thought processes, leading to varied evacuation intentions and actions. Therefore, understanding human psychology and behavior is essential in the design and development of effective evacuation models.

Vorst (2010) used John Leach's Dynamic Disaster Model which divided a disaster into three phases including the pre-impact phase, the impact phase, and the post-impact phase. Moreover, a total of five stages were developed: the threat stage, the warning stage in the pre-impact phase, the recoil stage, the rescue stage, and the post-traumatic stage in the post-impact phase. The model also suggested that specific human behaviors that are considered psychological responses to disasters are consistent throughout each stage and phase and do not vary greatly from disaster to disaster.

Vorst (2010) also examined the evacuation rate before and after a hurricane to explain the importance of taking human factors into account when developing evacuation models. It was found that prior to a disaster, approximately 30% of residents refused to evacuate. However, as the emergency became more urgent during the disaster, this number decreased to 5%. These behavioral responses contributed to a 25% increase in overall evacuation time. Additionally, the tendency of families or groups to remain

together during evacuation further impacted efficiency. The study also noted that women experienced higher levels of stress compared to men, which resulted in an average increase of 20% in evacuation time.

As previous research has rarely emphasized these aspects, Vorst (2010) advocated for the inclusion of psychological parameters and human behavioral factors in evacuation modeling. Integrating these elements can lead to more comprehensive and realistic simulations, enhancing the effectiveness of emergency response strategies.

Social Force (SF) Models

Social-force (SF) models for pedestrian dynamics were first introduced by Dirk Helbing and Péter Molnár in 1995. In their model, the motion and decisions of pedestrians can be described by social forces, an analogy from kinetic molecular motions, thus providing valuable information for planning and designing public area construction and evacuation (Helbing & Molnar, 1995). These 'social forces' are conceptualized as a set of invisible factors that influence pedestrian behavior, similar to the forces acting between molecules that govern their movement and interaction. Specifically, social forces in the context of pedestrian dynamics include the desire to reach a destination, the need to maintain a personal space bubble to avoid collisions with others, the tendency to align with the flow of surrounding people, and the impact of environmental elements such as barriers. Each of these forces plays a crucial role in shaping how individuals navigate through and interact with their surroundings, making the SF model a powerful tool for understanding and predicting pedestrian movement patterns in complex environments. They explained that the “forces” in the model were not simply exerted by the

environment in which pedestrians were located; instead, they were the measure of internal motivations for each individual to make a decision or perform a specific motion.

Human behavior is hard to predict and irregular, which is “chaotic” when there are multiple individuals who exist in one environment. Their motivations and motions are unpredictable and sometimes even interact with each other to make the situation more complex. However, when limited to relatively simple stochastic conditions, individual motion and behavioral models can still be developed by focusing on the probabilistic behaviors observed across large populations. As Helbing and Molnár (1995) noted, this approach allows for the modeling of pedestrian behavior based on aggregated behavioral probabilities, as exemplified in the gas-kinetic pedestrian model introduced by Helbing (1993).

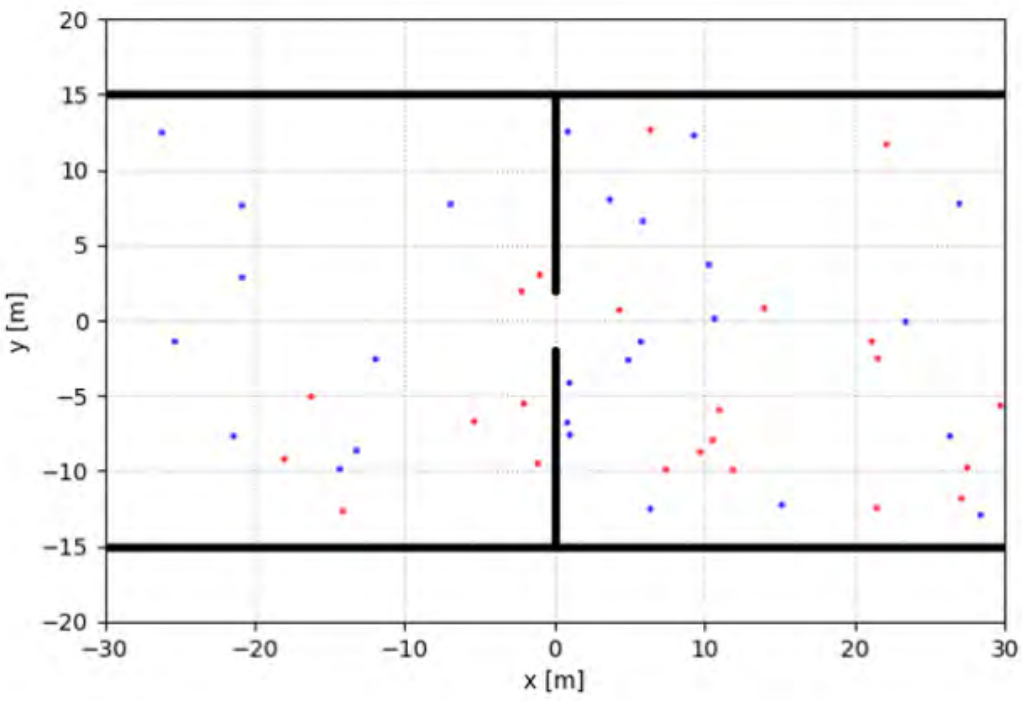

Fig. 6. Illustration of the pedestrian model

Figure 6 illustrates how the system and model simulate the action and decision-making process of pedestrians. In the illustration, pedestrians are divided into two different groups, with each color representing a motion direction. One pedestrian from the left has crossed the narrow door, which makes the other pedestrians in the same group tend to move forward with him/her in the same direction. This behavior can be explained by social forces: the pedestrian who crossed influences the others through a social force that encourages group members to follow, enhancing their tendency to move in the same direction. Consequently, pedestrians in the opposite direction then have to wait, a behavior driven by social forces that represent the psychological and physical need to avoid collisions and maintain personal space. This waiting behavior is a manifestation of repulsive social forces exerted by the moving group, which temporarily increases the social pressure on the opposing pedestrians to pause their movement and yield space. The diameter of each circle represents the speed of each individual, indicating how social forces not only influence direction and decision-making but also the speed at which pedestrians feel comfortable moving in response to the surrounding social dynamics.

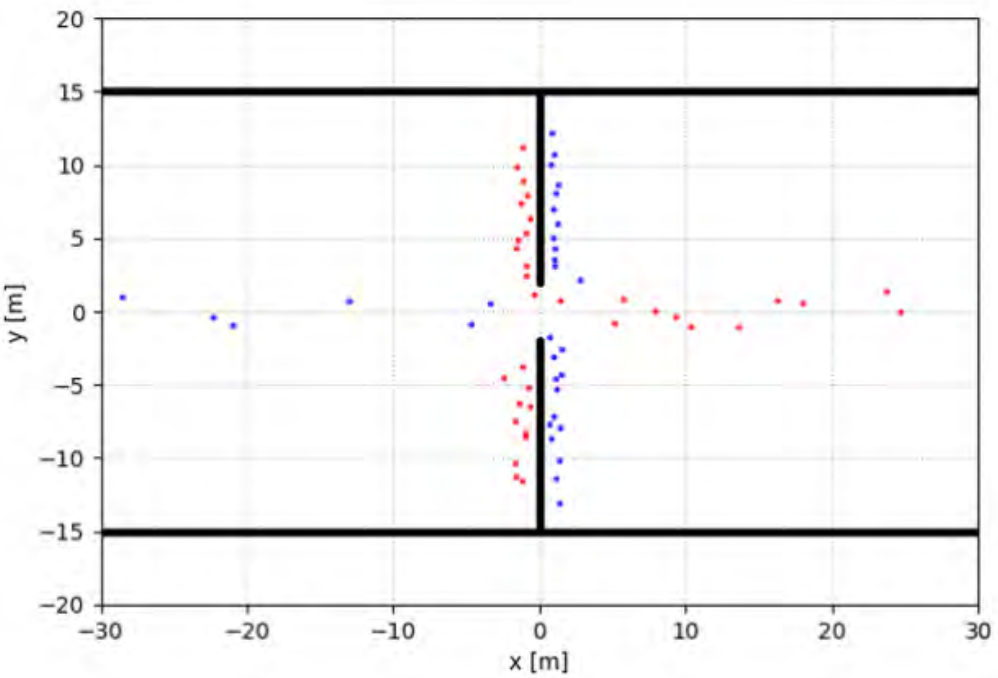

Fig. 7. Observation of pedestrian density and lanes formation

Another illustration (Figure 7) shows how the social force model operates and the factors that affect individuals' decisions. In the figure, there is the flow of pedestrians can be observed with a uniform moving direction when the density of individuals is above a certain point. The computational results were calculated and based on "N=4 (or 5) lanes on a walkway that is 10 m wide and 50 m long" (Helbing & Molnar, 1995). The pedestrian still moves in two opposite directions while there is no command or rule forcing them.

Zhang et al. (2021) implemented the social force model with deep reinforcement learning (DRL) algorithm and applied them to emergency evacuation in a room with obstacles . They found that although the social force model is fairly successful in simulating emergency evacuation situations, the optimal evacuation result is still questionable in complex environments that have obstacles. Therefore, Zhang et al. (2021)

developed the DRL algorithm with the social-force model, which helped train the agents to find the evacuation exits fast and efficiently.

In their study, Zhang et al. (2021) began by modeling an empty room with no obstacles to establish a baseline for evaluating evacuation performance. In this simplified environment, the self-driven force model directed agents straight to the nearest exit, allowing the researchers to validate its correctness. The results indicated that the median evacuation times for the two methods tested in the study were “not significantly different” (Zhang et al., 2021), confirming the model’s initial accuracy. Subsequently, the environment was modified to include a single obstacle and one exit. Under this condition, both their method and the traditional social force model produced comparable results. Building on this, the researchers extended their work to more complex scenarios involving multiple exits. The findings demonstrated that agents trained within the shared network were capable of successfully evacuating through the nearest exit. Each agent retained its own motivation and actions while benefiting from shared learning, showcasing the efficiency of the training process using the social force model. In summary, the integration of Q-learning and the social force model proved effective in managing emergency evacuations in environments with multiple exits and obstacles. The approach not only enabled agents to identify and reach the most efficient exit routes but also facilitated fast and safe evacuations in complex, dynamic settings.

According to Helbing and Molnar’s (1993) research, it is possible to view and predict individuals’ behavior into a set of equations of motion. When a pedestrian encounters a familiar or previously experienced situation, their response is typically guided by past experiences, prompting them to react automatically in a manner they

perceive as most appropriate. Thus, the systematic temporal "changes *dwa/dt* of the preferred velocity *wa(t)* of a pedestrian a are described by a velocity quantity *Fa(t)* that can be interpreted as a social force" (Helbing & Molnar, 1995). Although the force represents the effect from other individuals and the borders, the force is not actually applied to the individual's body. Instead, the social force represents the motivation of the pedestrian to take action. A pedestrian would only take action when he/she is subjected to external force (from other individuals or borders) in a social force model. This theory has been mathematically founded and adapted into research (Helbing, 1993).

Multi-Agent Reinforcement Learning (MARL)

According to the research done by Papoudakis et al. in 2020, a multi-agent reinforcement learning (MARL) system can be viewed as a "society of agents," where agents can interact with each other and cooperate to achieve a common goal (Papoudakis et al., 2020). Compared to traditional single-agent methods that are used, a multi-agent system has the advantage that could shorten the time of simulation and calculation while staying accurate. Because of the multiple number of agents, a complex task or exploration can be solved by jointly interacting agents with coordinated behaviors in a short period of time (Gosavi, 2004). However, developing a multi-agent system does not mean simply adding multiple numbers of agents into the same environment and making them work. Since the complexity of the entire system will increase because of the increased interaction and action of agents, the interaction between agents is more important than the number of agents.

In a cooperative environment of multiple agents, the policies and instructions for each different agent will become difficult to perform or optimize sometimes because of

the curse of dimensionality in the environment. In contrast, the delay between the correlated actions and rewards is unignorable and significant in the cooperative multi-agent environment. Because of the determined character of maximizing the gain for each different agent, there will be the agent who could not receive their deserved reward and been affected by other agents' favorable actions. Thus, to address the problems of the curse of dimensionality and the unbalanced rewards between multiple agents, the agents should be trained from a macro aspect, which helps the agents have a bigger view of the whole environment and system, thus improving the results and solutions for the multi-agent model.

Researchers have combined the advantage of reinforcement learning and multi-agent system and applied MARL in various fields. Martinez-Gil et al. (2011) utilized the MARL technique in learning the behaviors and navigational patterns of humans to simulate virtual pedestrians' movements. They managed to reveal the basic movements and real characteristics of pedestrians, which indicates the validity of the MARL algorithm and model in simulating multiple agents' movements. In the paper, they prepared two reinforcement learning algorithms based on Vector Quantization (VQ) for Q-Learning (VQQL) algorithm (Martinez-Gil et al., 2011). The results indicated that both approaches are able to obtain adequate vector quantization for each different agent in the environment. MARL technique can be used to solve a large amount of agents' movement problems like evacuation under emergency situations. With the help of reinforcement learning, the system is able to scale movement path and speed control, which is extremely helpful when emergency situations occur.

Emergencies that occur in small, enclosed spaces with high population densities, such as classrooms and movie theaters, are often the most likely to cause casualties. Thus, evacuation designs and plans are becoming increasingly important. Since it is hard for static evacuation methods to demonstrate the difference in safety for all alternative designs, and putting people in real emergencies is time-consuming, dangerous, and expensive, Liu et al. (2016) developed an agent-based simulation model via NetLogo to investigate the correlation between evacuation efficiency and classroom layout. This model, utilizing the principles of multi-agent systems (MAS), simulates the complex interactions between individual agents representing people during an evacuation. By establishing a set of behavioral rules for these agents based on real-world experience, the agent-based modeling (ABM) techniques employed in the study can successfully reflect the dynamic characteristics of human evacuation behaviors. Each agent operates autonomously yet interacts with other agents and the environment in a manner that captures the collective behavior of crowds in emergency situations, showcasing the power of MAS in understanding and optimizing evacuation processes.

There are four major parts in the simulated environment of Netlogo: turtles (agents), patches (grids that constitute a 2-D world), links (the social and physical connection between agents), and observers (supervisors). Two major types of general classroom layouts were used, while two different types of evacuation scenarios (a self-organized scene and a premeditated scene) were designed and thus led to four categories. The self-organized agents only are set as selfish and independent and only consider running out quickly so that the population density and distance to the nearest exit would most influence their decision-making process. In contrast, the premeditated agents who

have conducted fire drills would follow the instructions and evacuate while avoiding collision with other agents. Speed limits were also introduced into consideration to get more realistic results. Agents were randomly divided into three groups that had a speed of 0.6 m/s, 1m/s, and 1.2 m/s. The results indicated that a classroom layout with two exits would shorten evacuation time while the agents under premeditated behavior rules could evacuate both quick and safe.

Summary

This review of existing literature on emergency situations, aviation emergencies, evacuation processes, and technological approaches such as machine learning, reinforcement learning including the Asynchronous Advantage Actor-Critic (A3C) algorithm, social force models, and multi-agent reinforcement learning algorithms highlights the significant progress made in this field. However, there is still a noticeable research gap in the practical application and effectiveness of advanced computational models, specifically on utilizing the A3C algorithm in real-world evacuation scenarios. Most studies have concentrated on theoretical or simulated environments with limited exploration into complex and real-life emergency situations. These situations normally involve dynamic threats and a high density of individuals. This study aims to bridge this gap by not only applying the A3C algorithm in realistic evacuation simulations but also by comparing its efficacy with established models like DQN. The findings from this research enhances understanding of the practicality and adaptability of reinforcement learning algorithms in emergency evacuations, which is crucial in aviation emergencies, where timely and efficient evacuation can significantly impact safety and survival. By advancing knowledge in this field, this study contributes to the development of more

effective evacuation strategies, potentially saving lives and optimizing emergency responses in high-risk situations.

Section 3: Methodology

The model environment used in this study were built using two types of algorithms: A3C and DQN. This section provides a detailed description and explanation of these two algorithms, including the models, code structure, and how they were applied to the model environment.

A3C algorithm

The model's primary goal was to achieve efficient and optimal decision-making for evacuation during emergency situations using the A3C algorithm. As stated in the literature review section, the Asynchronous Advantage Actor-Critic (A3C) is a powerful reinforcement learning algorithm that addresses the challenges associated with traditional reinforcement learning methods. One of the main benefits of A3C is its ability to use multiple actors to explore different parts of the environment in parallel simultaneously. These speeds up the training and learning process and reduces the redundancies between agents and training environments (see Figure 8).

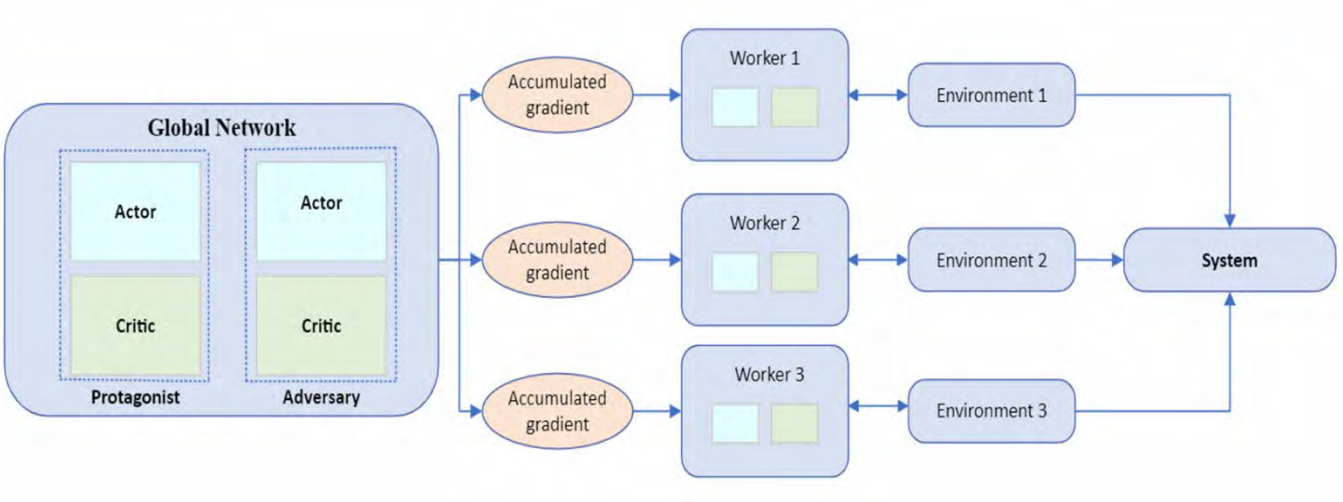

Figure 8. Asynchronous Advantage Actor-Critic Network

The A3C algorithm-based model consists of two components that achieve the Actor-Critic structure: the actor and the critic. The actor represents the policy function for the agents in the environment, defining the probability of each step for each agent based on the current state of the environment. One essential aspect of A3C is instructing and guiding the agents' actions for the best possible outcome. In contrast, the critic plays an important role in estimating the value function of the current policy for the agents, by evaluating the quality of the actor's actions.

The critic computes the advantage score and updates the policy for the actor to obtain a better policy and rewards in the following episodes. By combining the advantages and functions of both actor and critic, A3C can utilize both the benefits of policy-based and value-based approaches to achieve best learning efficiency. The policy-based approach directly optimizes the policy function that maps states to actions, enabling more flexible behavior learning. In contrast, the value-based approach is generally more stable and exhibits lower variance in its estimates, though it may be less effective in high-dimensional or continuous action spaces (Sutton and Barto, 2018).

Therefore, A3C combines the strengths of both methods and makes it more effective in a wide range of RL problems, including those with complex action spaces.

The following pseudocode illustrates the logic for A3C algorithm and how each actor-learner thread works (Mnih, 2016). The θ represents the policy in this algorithm, which was strictly followed by the agent for taking their actions and making decisions. As θv models the value function V, it evaluates the effectiveness of the action that agents took. Both θ and θv are implemented using convolutional neural networks (CNNs) with shared parameters, except for the output layer. In order to output the probabilities of actions for the agents to take, a softmax function was given to θ. For θv, a linear layer was used for outputting a scalar value for evaluation of the actions.

Algorithm 1 Parallelized Actor-Critic with Advantage Estimates

---

// Define global shared vectors θ (policy) and θ_v (value), and a global counter $\mathcal{T} = 0$
// Define thread-specific vectors θ' (policy) and θ'_v (value)
Initialize thread counter: $t \leftarrow 1$
repeat
    Reset gradients: $d\theta \leftarrow 0$, $d\theta_v \leftarrow 0$
    Align thread-specific parameters: $\theta' \leftarrow \theta$, $\theta'_v \leftarrow \theta_v$
    Set thread iteration start: $t_{start} \leftarrow t$
    Obtain current state: s_t
    repeat
        Execute action a_t as per policy $\pi(a_t|s_t; \theta')$
        Receive reward r_t and observe new state s_{t+1}
        Increment thread counter: $t \leftarrow t + 1$
        Increment global counter: $\mathcal{T} \leftarrow \mathcal{T} + 1$
    until s_t is terminal or $t - t_{start} = t_{max}$
    Compute R for state s_t: $R = 0$ if s_t is terminal, or $R = V(s_t, \theta'_v)$ otherwise // Bootstrapping for non-terminal states
    for each step i from $t - 1$ to t_{start} **do**
        Update R: $R \leftarrow r_i + \gamma R$
        Calculate gradient for θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i; \theta')(R - V(s_i; \theta'_v))$
        Calculate gradient for θ'_v: $d\theta_v \leftarrow d\theta_v + \partial (R - V(s_i; \theta'_v))^2 / \partial \theta'_v$
    end for
    Apply asynchronous updates to θ and θ_v using $d\theta$ and $d\theta_v$
until $\mathcal{T} > \mathcal{T}_{max}$

Starting from initialization, the Global shared counter *T* was initialized to 0, while the thread step counter was initialized to 1. The parameter vectors in this algorithm were also set as two types: global shared parameter vectors θ and θv, and the thread-specific parameter vectors θ' and $\theta' v$. At each step, the algorithm updated the thread-specific parameter vectors θ' and $\theta' v$ to match the global parameter vectors θ and θv. The system then interacts with the environment, where the action a_t is chosen based on the current state *St* and the policy $\pi(a_t|St\ ;\ \theta')$. It observes the reward *rt* and the new state *St+1*. This step repeats until a terminal state is reached or a maximum number of steps, which is $t - t_{start}$ are taken. When the algorithm encounters a terminal state, indicating the end of an episode, the expected future return R is set to 0, as there are no further rewards to be obtained beyond this point. In case of if the final state is non-terminal, the return will be generated using the value function $R = V(st, \theta' v)$. The return R and accumulated gradients with respect to θ' and $\theta' v$ are calculated for each step from *t-1* to t_{start}. The return is then calculated by summing up the rewards from the current state to the end of the episode, with each reward discounted by a factor of γ raised to the power of the number of time steps away from the current state:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{T-t} r_T$$

Where r_t is the reward at time t and γ is the discount factor. The γ, ranging between 0 and 1 determines the importance of future rewards. A value of $\gamma = 0$ means the agent considers only immediate rewards, while a value close to 1 indicates that future rewards are given greater importance. Thus, in this study, the γ was set to 0.99, which is a relatively large gamma value in reinforcement learning applications, since future rewards are considered almost as important as the immediate rewards in the case of successful

evacuations in the airport. A discount factor close to 1 encourages agents to prioritize long-term rewards, leading them to select actions that yield better outcomes over time. For a stochastic dynamics environment like an airport, a higher discount factor could also help stabilize the learning process and reduce the variance in the updates. Although the training and learning process could be slower due to the large value of discount factor, 0.99 is still a good gamma value after hyperparameter tunning process.

The A3C algorithm operates within the Policy Gradient framework and typically employs the Temporal Difference (TD) error as a critic to evaluate the performance of the actor. The following formula represents the empirical estimate of the policy gradient for Reinforcement Learning problems:

$$\nabla_\theta R_\theta \approx \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\left(\sum_{t'=t}^{T_n}\gamma^{t'-t}r_{t'}^{n} - b\right)\nabla_\theta \log \pi_\theta\left(a_t^n|s_t^n\right)$$

As a policy gradient method, A3C improves the stability of the method and convergence properties of policy gradient methods. The advantage function, used in A3C, measures the difference between the expected return of taking an action in a particular state and the expected return of behaving according to the current policy in that state. The advantage function can be written as:

$$Advantage\ (A_t) = R_t - V(s_t)$$

In this simplified formula, R_t represents the cumulative discounted rewards from time to time. Meanwhile, $V(s_t)$ represents the critic part, which mainly provides estimation of the state value for the state s_t. The advantage (A_t) essentially measures how much better or worse an action is compared to the expected value estimated by the critic. To be more specific, the advantage formula can be detailed as follow:

$$A(s_t, a_t) = r_t + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$

Here, the r_t is the reward at time t. As previously mentioned, *V(st)* is the value of state s_t estimated by the critic and γ is the discount factor. By using the advantage function, the algorithm calculates the difference between the cumulative discounted rewards and the expected value provided by the critic, allowing it to assess how much better an action truly is.

As the advantage formula gauges the expected return disparity between the action and current policy, the update formula is also essential for calculating the expected reward gradient in respect to the policy parameters. The update A3C policy optimization formula can be expressed as follow:

$$\nabla_{\theta} J(\theta) \approx \nabla_{\theta} \log \pi_{\theta} (a_t|s_t) A(s_t, a_t)$$

$\pi_{\theta}(a_t|s_t)$ is the probability of taking action at in state s_t under policy θ. The update formula is essentially computing the gradient of the expected reward with respect to the policy parameters, scaled by the advantage.

In practice, the A3C algorithm also introduces an entropy regularization term to encourage exploration during the training process. By encouraging the agents to choose something without a curtain decision or knowledge (exploration), the agents learn more from the environment and the policy by interacting more, which is a fundamental dilemma in reinforcement learning (Figure 10).

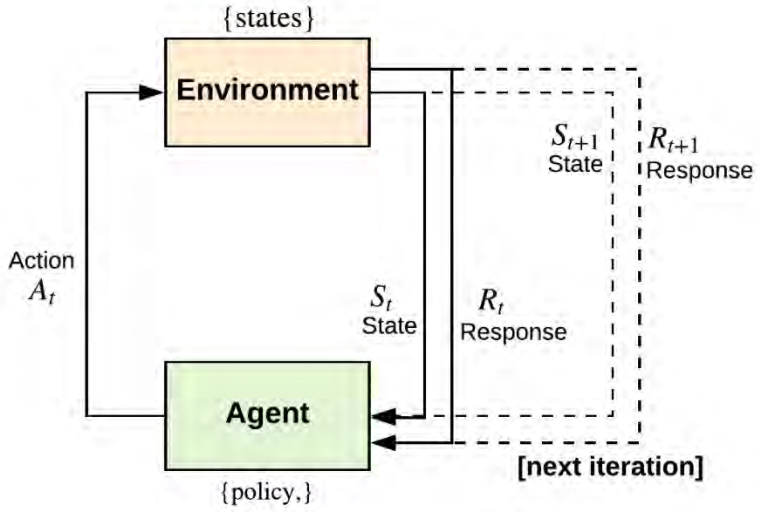

Figure 10. The Exploration-exploitation Trade-off

After adding the consideration of the exploration-exploitation trade-off, the total loss for the A3C algorithm typically consists of three parts: the policy gradient loss, the value function loss, and the entropy regularization. The full objective function that A3C tries to optimize can be formulated as:

$$L(\theta) = -\log \pi_\theta (a_t|s_t) A(s_t, a_t) + \lambda \left(V(s_t) - \hat{V}(s_t)\right)^2 - \beta \sum_a \pi_\theta(a|s_t) \log \pi_\theta (a|s_t)$$

The first term, $-\log \pi_\theta (a_t|s_t) A(s_t, a_t)$, represents the policy gradient loss. This term encourages the probability of actions that lead to higher-than-expected returns to increase, and the probability of actions that lead to lower-than-expected returns to decrease. Here, $A(s_t, a_t)$ is the advantage function, which measures how much better an action is compared to the average action at state S_t. The second term, $\lambda \left(V(s_t) - \hat{V}(s_t)\right)^2$, is the value function loss, which is a mean-squared error that penalizes the agent for incorrect value estimates. The agent's current estimate of the state value $V(s_t)$ is compared to the target value $\hat{V}(s_t)$, which is computed using the bootstrapped returns.

This value function loss ensures that the value function is a good predictor of future rewards, which is critical for calculating accurate advantage estimates. Finally, the entropy regularization $-\beta \sum_a \pi_\theta(a|s_t) \log \pi_\theta(a|s_t)$ encourages exploratory behavior by dissuading the policy from becoming overly deterministic, thus maintaining a healthy balance between exploration and exploitation. As the training process minimizes these three terms of the loss function, it concurrently refines the parameters of both the policy and the value function. This refinement guides the agent towards a more effective strategy over successive iterations.

DQN Algorithm

To compare efficiency in terms of time and learning curves by measuring the average reward, the Q-learning algorithm was employed as a foundational component of the Deep Q-Network (DQN) to simulate evacuation procedures in an airport environment. DQN served primarily as a standard baseline for this study because it is commonly used in many types of simulations, including evacuation scenarios, within reinforcement learning studies. As one of the earliest deep learning-based reinforcement learning algorithms, DQN successfully demonstrated that neural networks could approximate Q-values for high-dimensional state spaces. This capability makes it well-suited for the complex simulations of this research and establishes a solid performance benchmark for comparison with the A3C algorithm.

Q-learning is an reinforcement learning algorithm that seeks to learn the value of an action in a particular state, aiming to maximize the total reward. The pseudocode provided (Figure 11) offers a clear and basic illustration for the implementation of the Q-learning process (Yang, 2020).

Algorithm 2 Q-learning Algorithm

1: Initialize $Q(s, a)$ as zero;
2: **for** each episode **do**
3: Initialize state s
4: **for** each step of the episode **do**
5: Choose action a from state s using Q-table
6: Take action a, observe r, s'
7: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
8: $s \leftarrow s'$
9: **end for**
10: **end for**
11: **until** s is terminal

Pseudocode of the Q-learning algorithm is the foundational component of DQN The Q-values are initialized arbitrarily, providing a starting point for the algorithm to begin learning. These values represent the expected utility of taking a given action in each state and are updated iteratively as the agent interacts with the environment. This initialization provides a baseline from which the algorithm can begin the learning process. Then the learning occurs over a series of episodes. Each episode begins with the initialization of the states (s), representing the starting conditions of the evacuation simulation, such as the initial position of agents within the airport. Within each episode, actions are chosen using an ε-greedy policy, which involves selecting the action with the highest Q-value (exploitation) with probability $1-\epsilon$ or a random action (exploration) with probability ϵ:

$$With\ probability\ 1\ -\ \varepsilon: Choose\ a\ =\ argmax\_a\ Q(s,a)$$

$$With\ probability\ \varepsilon: Choose\ a\ randomly$$

After executing an action, the Q-value is updated based on the Bellman equation, which incorporates the immediate reward and the discounted maximum future reward:

$$Q(s,a) \leftarrow\ Q(s,a) +\ \alpha\ [\ r\ +\ \gamma \max(over\ a')\ Q(s',a') -\ Q(s,a)]$$

In the DQN algorithm, $Q(s, a)$ represents the current estimated value of taking action (α) in state (s), encapsulating the expected utility of this decision. The learning rate, denoted as (α), plays a critical role in balancing new incoming information with existing knowledge, controlling the rate at which the algorithm adapts. The immediate reward received from taking action (α) in state (s) is represented by (r), serving as a direct feedback mechanism for the action's effectiveness. The discount factor, γ, quantifies the importance of future rewards, enabling the algorithm to prioritize short-term gains against long-term benefits. Upon acting (α), the agent transitions to a new state s', with $\max(over\ a')\ Q(s', a')$ indicating the best possible future reward attainable from this new state, guiding strategic decisions towards the most beneficial outcomes. This update process is repeated for each step within an episode, continuing until a terminal state is reached. This marks the conclusion of one simulation run, with the goal of optimizing evacuation strategies in the face of dynamic challenges and threats.

The DQN algorithm improves the classical Q-learning framework by incorporating a deep neural network to estimate the Q-value function. This is crucial for handling the high-dimensional state spaces found in complex environments such as airports. The algorithm strategically uses replay memory to store transitions experienced during the simulation, which are later randomly sampled to update the neural network. This method diversifies the learning experience, preventing overfitting to recent transitions and smoothing the training process over numerous episodes. As a result, the DQN algorithm can incrementally refine its Q-values and gradually identify optimal evacuation paths. The algorithm demonstrates an ability to learn and improve evacuation

strategies iteratively by confronting and negotiating various challenges, such as dynamic threats and the coordination of multiple agents.

The utilization of the DQN algorithm, supported by an advanced Q-learning framework, enables effective simulation and enhancement of airport evacuation procedures. The algorithm's adept handling of complex scenarios through continuous learning and adaptation illustrates its potential in optimizing evacuation paths, ultimately contributing to safer and more efficient emergency responses. In this study, DQN is not only a powerful tool but also a critical benchmark for evaluating the A3C algorithm. By comparing the performance and learning dynamics of DQN and A3C within the same simulation environment, the strengths and limitations of each approach can be clearly delineated. This comparative analysis enhances comprehension into the applicability of reinforcement learning algorithms to real-world challenges, such as airport evacuations. It positions DQN as a fundamental model against which the innovations and improvements of A3C can be measured, highlighting areas where A3C may offer advantages in terms of learning efficiency, scalability, and adaptability to complex, dynamic environments.

Environment

The environment utilized in this project was developed and customized solely using the Gym toolkit (Brockman et al., 2016). Developed by OpenAI, Gym was designed as a toolkit for creating and comparing various types of reinforcement learning algorithms. As the agents operate under the policies of different algorithms within the Gym environment, they generate varying average rewards and levels of efficiency, culminating in unique learning results. The environments available through Gym span from traditional control tasks and games, such as Atari, to robotic simulations. In this

particular study, a unique Gym environment was developed from scratch, simulating a real airport environment, without dependence on pre-existing environments. The customized environment permits agents to function across any algorithms, evading limitations to a specific algorithm. The customized environment created by Gym is ideal for the project's objective of fitting agents and enabling them to find the optimal evacuation path under various algorithms in different airports. Gym's capabilities allow the transformation of any real-life airport into a virtual environment, where agents can determine the best evacuation route. In other words, the biggest advantage for using Gym environment is that it combined with the self-learning feature of machine learning, generates the optimal route for any actual airports, thereby saving time and lives.

This project's customized environment consists of several key components: environment space, agent, observation, reward, and episode. The environment space is where agents move and take actions, defined by a state space and an action space. The state space is the system's current state, as indicated by the agent location, which will be clearly defined in the code. The action space encompasses all nine potential actions an agent can take within the environment: the four cardinal directions (North, South, East, West), the four intercardinal or ordinal directions (Northeast, Southeast, Southwest, Northwest), and remain static. The environment's comprehensiveness and detail are enhanced by these two spaces. The agent is a crucial element within the environment, interacting by taking actions within specific states. When the agent takes action, it receives observations and rewards from the environment, which helps it make better decisions in future episodes based on its policy and learning algorithms. The environment provides agents with observations and rewards, which are information about the current

state of the system and scalar feedback based on the agent's actions. By utilizing these fundamental elements, agents can continue to take actions and maximize their cumulative reward over time.

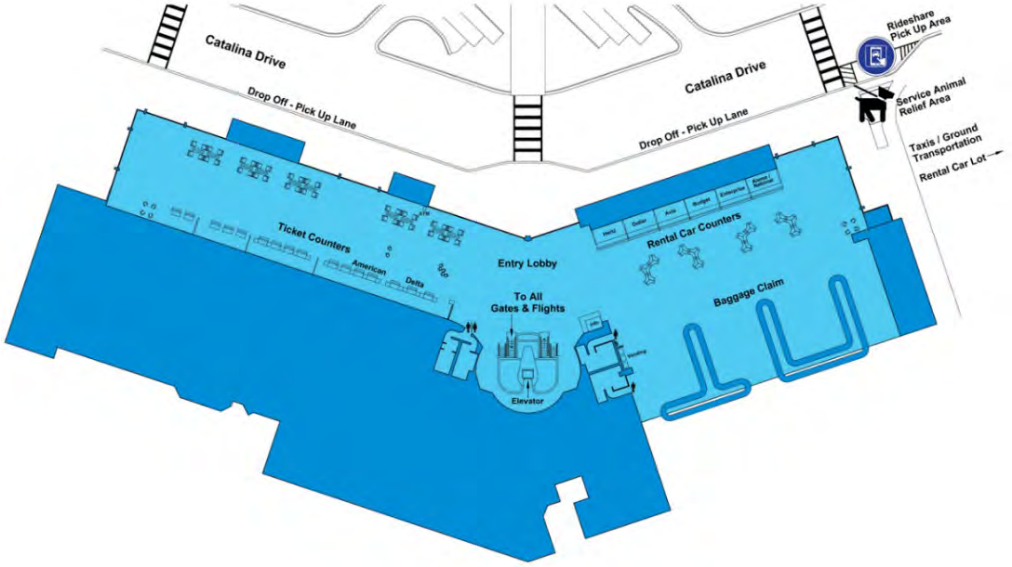

Figure 12. A Local Airport Ground Floor Map

The customized environment was created by transforming a pre-existing airport map. The map was based on a real-world local airport to ensure a more accurate and comprehensive result for emergency evacuation (see Figure 12). Adopting the real-world airport map into the reinforcement learning environment enabled advantages such as increased realism, complexity, and detail. The realistic and complex environment of real-world airports, in contrast to the simplified pre-built environments like those in Gym, supports the development and training of more robust and capable algorithms such as A3C and DQN. These algorithms are better suited to handle a wide range of dynamic and unpredictable scenarios. Furthermore, a simulation environment modeled after a local airport can serve as a standardized benchmark, providing a consistent foundation for evaluating and comparing the performance of various algorithms under identical conditions.

The normal airport operating process consists of four phases: check-in, security, waiting area, and boarding (Figure 13). These phases typically occur at different locations within the airport, which may span multiple floors. This study focused on the first floor of the airport, where the check-in and security phases take place. Compared to the first two phases, there is less probability of emergency situations occurring in the waiting and boarding areas/phases. This is because passengers have already passed the security check and are carrying baggage and items that strictly follow the rules and restrictions of the airport and the Transportation Security Administration (TSA). Emergencies, such as bomb threats and fires, tend to occur more frequently during the first two phases: check-in and security, which is typically on the ground floor of the airport.

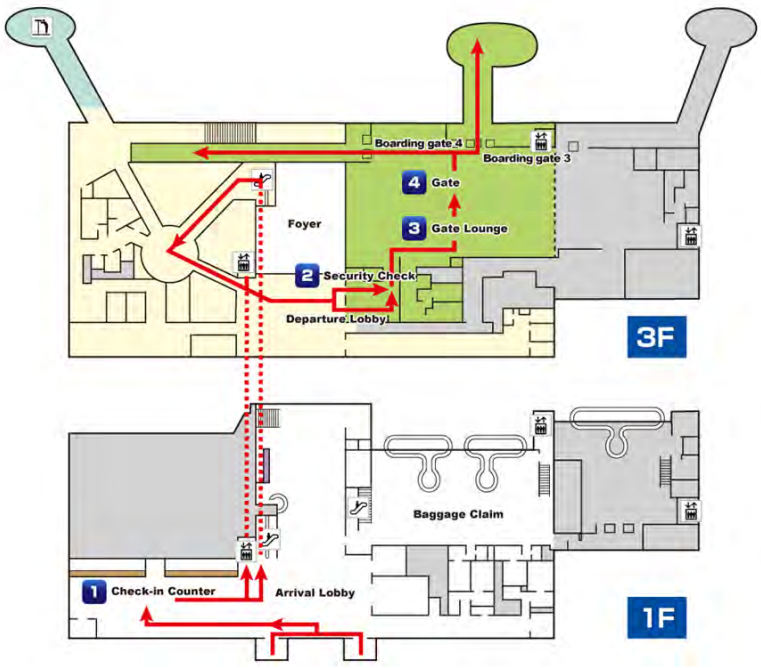

Figure 13. The Four Phases of Airport Operation

To accurately simulate the local airport map, ‘self.walls’ were defined in the customized environment to draw the outline as well as all of the obstacles in the airport. The line ‘self.walls = np.full((self.size, self.size), False)’ was used to create a 2D array

(or matrix) with the shape defined by '(self.size, self.size)'. Each element of this array was initially set to False to simulate the existence of the walls and obstacles in the airport. When agents in the model act, each of them will normally take one step in each direction it choses. For walking distance, the step length on average is 2.2 feet for a woman and 2.5 feet for a man (StepsApp, 2023). In emergency situations, passengers tend to run or move faster than walking, which increases the distance they cover during evacuation. Therefore, the size value for the map was set to 50, making the model environment scale to 2500 (50x50) pixels large. This scale provided the model with a large and realistic environment for the agent to explore and learn through iterative interactions. With the consideration of step length, obstacles in the airport, and compatibility of the Gym customized environment, the complete environment was built as shown in Figure 14.

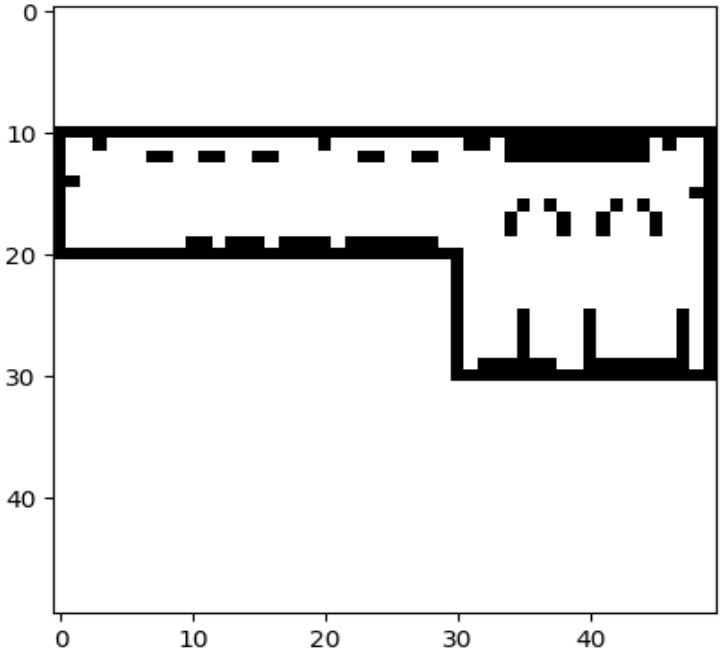

Figure 14. The Customized Environment

Due to Gym's compatibility constraints, instead of designing the map with angled layouts, the airport ground floor was divided into two sections, left and right, and combined them into a simplified rectangular layout. This approach eliminates additional

influencing factors and excludes extra rooms that provide access to the second floor, resulting in a more controlled environment for simulation.

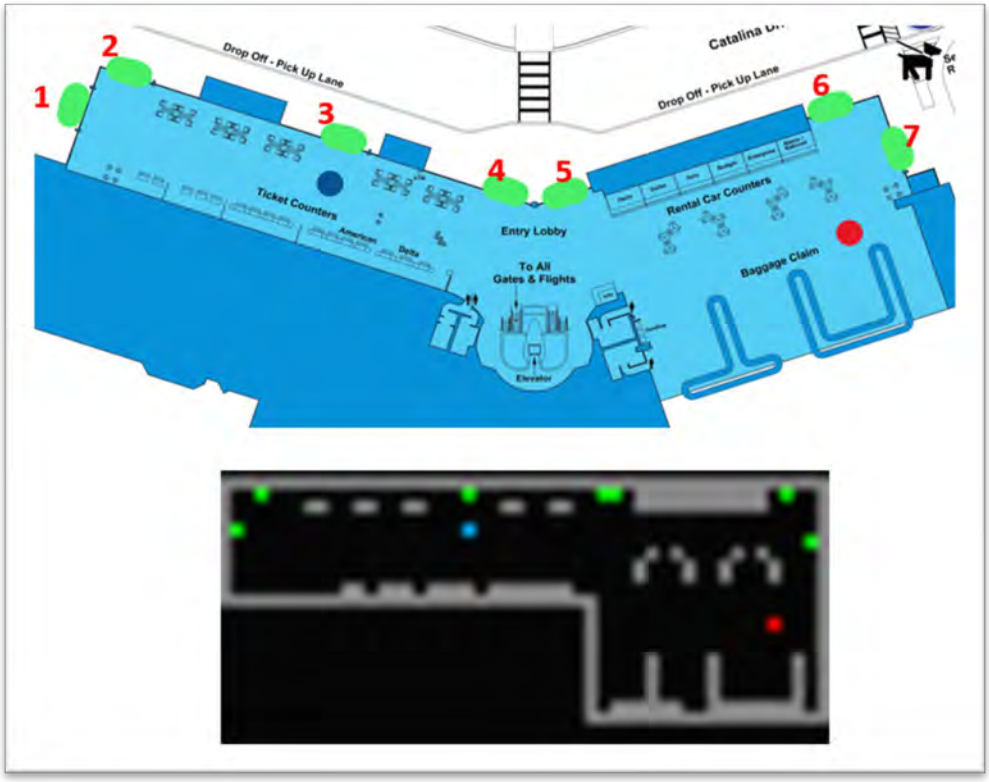

Figure 15. The Customized Environment by Adopting the Real-world Airport Map

In the model environment, the ticket counters, rental car counters, and baggage claim were kept to provide a realistic simulation of the airport. With consideration of the first two phases, check-in and security, the threat was placed near the baggage claim area, representing a potential fire accident or bomb threat. Therefore, there were four situations based on the model environment settings and number of agents, which were: single-agent static threat situation, single-agent moving threat situation, multi-agent static threat situation, and multi-agent moving threat situation. Each of the situations was test ed and used as the benchmark platform for both A3C and DQN algorithms in the project.

Static Threat Environment

In this scenario, a single agent was tasked with finding the shortest, fastest, and safest route to escape from the airport under the assumption of a static threat. Without the obstruction of other agents (passengers), the individual (single-agent) was free to explore the environment with each step taken. The experience gained was collected and learned from through this exploration-exploitation process using the algorithms. The scenario involved a bomb threat at the airport, which poses a danger to thousands of lives. The bomb threat was represented by a red dot that remained static while the agents explored the optimal path to escape from the airport. Figure 16 illustrates the complete scenario environment, including the single agent and bomb threat placement.

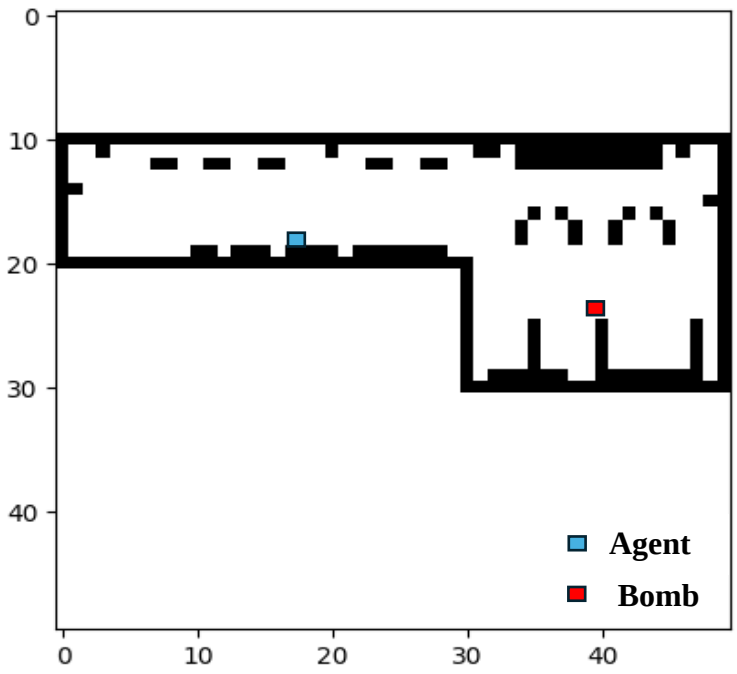

Figure 16. Single Agent Static Threat Environment

Moving Threat Environment

In the moving threat environment scenario, the limitation of having only one agent remains unchanged. However, the previous bomb threat was replaced with a more realistic and hazardous element: fire. Emergency situations, such as fire hazards, differ in scale and severity from other hazards and emergencies. As previously stated, while proper adherence to normal operations can mitigate most emergency situations to some extent, reactive measures such as evacuation plans and real-time decision-making systems play a crucial role in effectively responding to emergencies. Therefore, in the event of a sudden fire at the airport, emergency response and evacuation plans can greatly mitigate or resolve the situation without causing significant disruption to normal operations.

In this scenario, the threat was simulated using a fire spreading model developed by Martin (2023) in MATLAB, designed to follow a specific pattern that mimics real-world fire behavior. The original model was intended to simulate forest fires, generating a random forest on an n x n grid. A key feature of the model is the probability (p) of igniting an unburned tree located within a burning tree's Moore neighborhood, introducing stochastic behavior into the fire's spread. This randomness added complexity and realism to the simulation, making the environment more dynamic and unpredictable (see Figure 17).

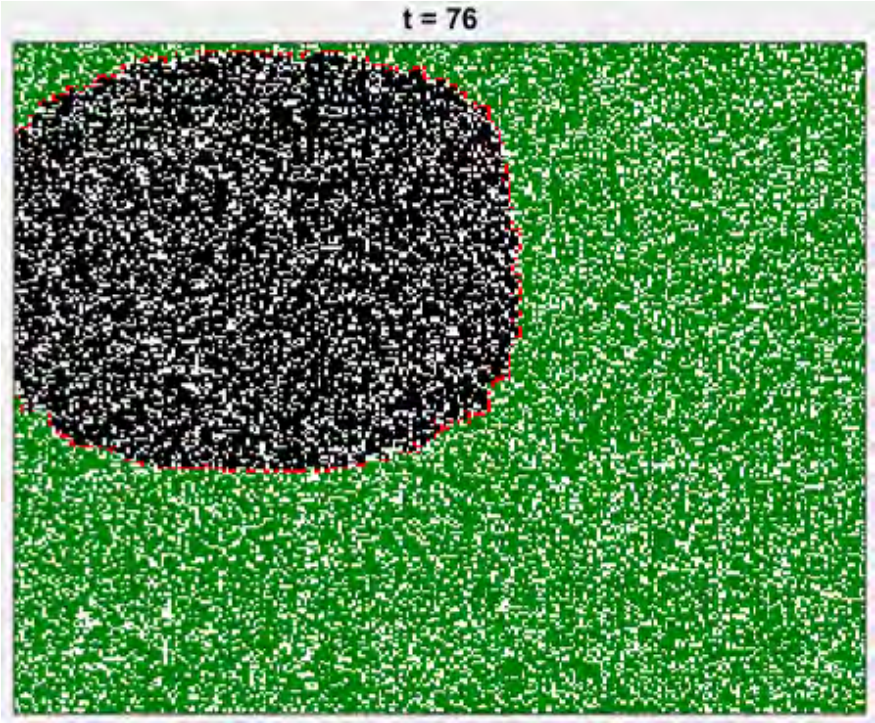

Figure 17. Simulation of a Forest Fire with the Spreading Model in MATLAB (Martin, 2023)

To enhance the accuracy of the model and make it more representative of fire spreading patterns in airport environments, the probability of igniting a tree was modified to represent the probability of spreading to the nearest cell and expanding in size. Unlike the original forest fire model, where fire spread depended on the presence of a 'tree' in the Moore neighborhood, the updated model allows the fire to spread regardless of cell content. This adjustment creates a more aggressive and realistic simulation of fire behavior in open, structured environments like airports. As a result, the fire spread model in this study became more realistic and quantifiable with each step the agent takes and learns (Figure 18).

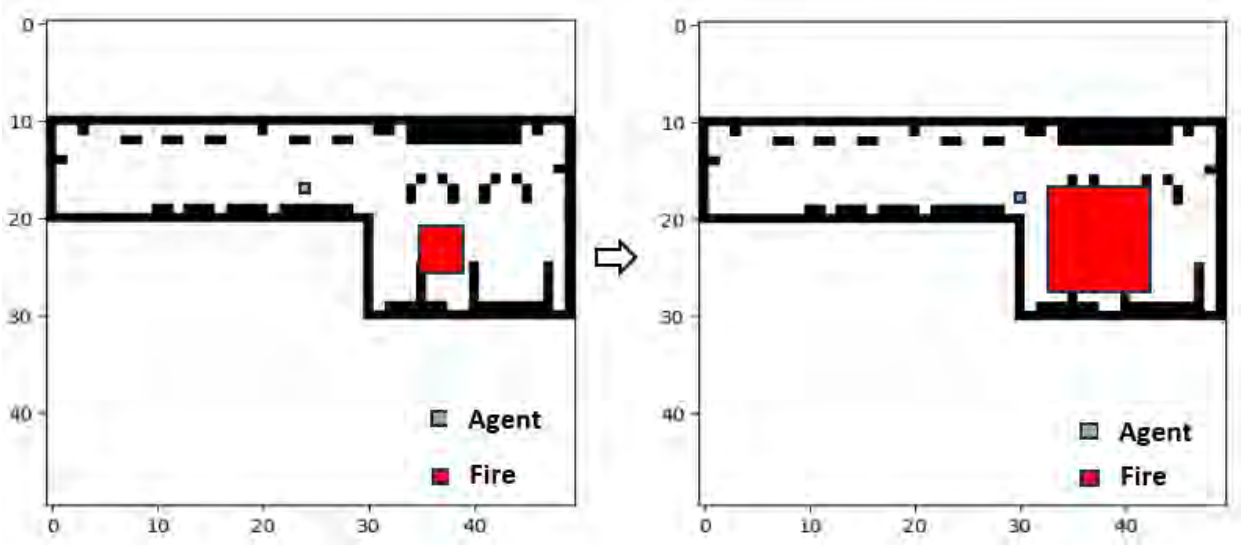

Figure 18. Single-Agent Fire Environment Changes with Episodes

The model was run on a computer equipped with an AMD Ryzen 5600X 3D CPU, an NVIDIA GeForce RTX 3060 Ti GPU, and 16GB of RAM. This setup provided the necessary computational resources for the demands of the reinforcement learning algorithms used in the study. Each model and simulation were conducted over 1000 episodes, with each episode representing a complete run of the environment from start to finish. The end result could have been one of three options including, successfully escaping from the airport, a collision with the static or moving threats, or even a crash between other agents in a multi-agent situation. To ensure statistical significance and account for variability, each model was tested across at least 50 trials. Throughout these episodes, key data points were collected, including metrics such as cumulative rewards per episode, the frequency with which specific exits were taken, and the duration of each episode from start to finish. The cumulative rewards per episode served as the basis for a comprehensive analysis of the models' performance in terms of each algorithm's learning

process. To facilitate a direct comparison between the two models under investigation, identical conditions were maintained across trials, and the same sets of metrics were used for evaluation. The comparison of learning curve and efficiency measured in seconds focused not only on the efficiency and effectiveness of the models in achieving the set objectives but also on their learning behavior over time and adaptability to the environment.

Rewards

Initially, the default penalty for an agent's contact with a threat was set at -20, aimed at encouraging the agent to focus more on learning the optimal path while avoiding threats. Setting an appropriate value for the threat penalty helped the agent learn to avoid threats more effectively, finding the optimal path more swiftly and safely. In order to find the proper value, various penalty values were explored to assess the impact of negative rewards on the learning curve of the A3C algorithm and to optimize overall performance by fine-tuning these reward values. Figure 19 below depicts the learning curves for two different threat penalty settings applied to the A3C model in a static environment:

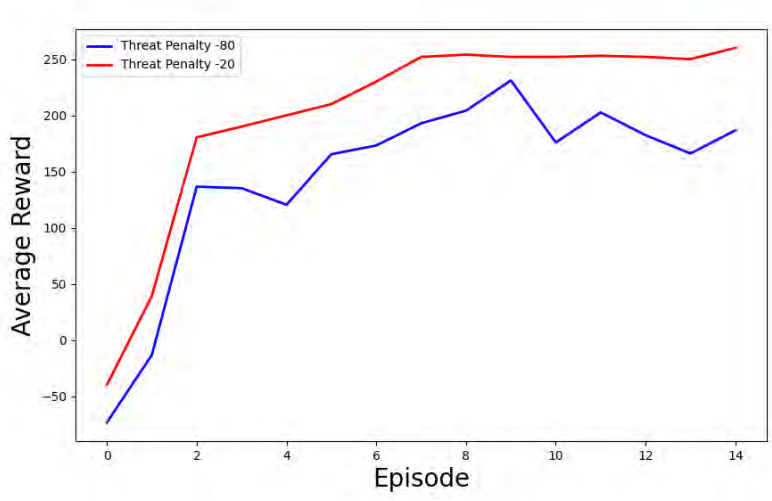

Figure. 19 Reward plot for A3C agent with different penalty, static threat

Figure 19 displays two different penalty values: -20 and -80. As seen from the graph, the learning curve for the -80 penalty exhibits significantly more fluctuation compared to the -20 penalty curve. This is because the agent incurs a higher penalty (-80) for contact with threats during the learning process, resulting in a considerably lower starting point for the average reward in the -80 setting compared to the -20 setting. The model with the -20 penalty appears to converge around 200 episodes, whereas the model with the -80 penalty does not converge and continues to learn, requiring additional episodes and time to determine the optimal path for evacuation. This observation indicates that for this environment, the A3C model may perform better with a less severe negative reward for threat penalties.

The comparative analysis of penalty settings was extended to the A3C model in a dynamic threat environment, as seen in Figure 20. Both learning curves exhibited increased fluctuations due to the presence of moving threats, consistently resulting in diminished rewards each time the agent encountered a threat. These findings suggest that the magnitude of the threat penalty can substantially influence the agent's movement and decision-making processes. When the penalty for encountering a threat is excessively high, the agent's primary focus may shift towards avoidance of the threat rather than navigating out of the hazardous environment. Consequently, agents might adopt an overly cautious strategy, eschewing optimal paths or actions associated with any risk of threat encounters, even if such risks could potentially yield greater rewards. This behavior could adversely affect the balance between exploration and exploitation within the reinforcement learning framework. An overly high threat penalty could discourage

exploration, impeding robust learning and leading the agent to prioritize penalty avoidance over the pursuit of potentially higher-reward paths that entail some risk.

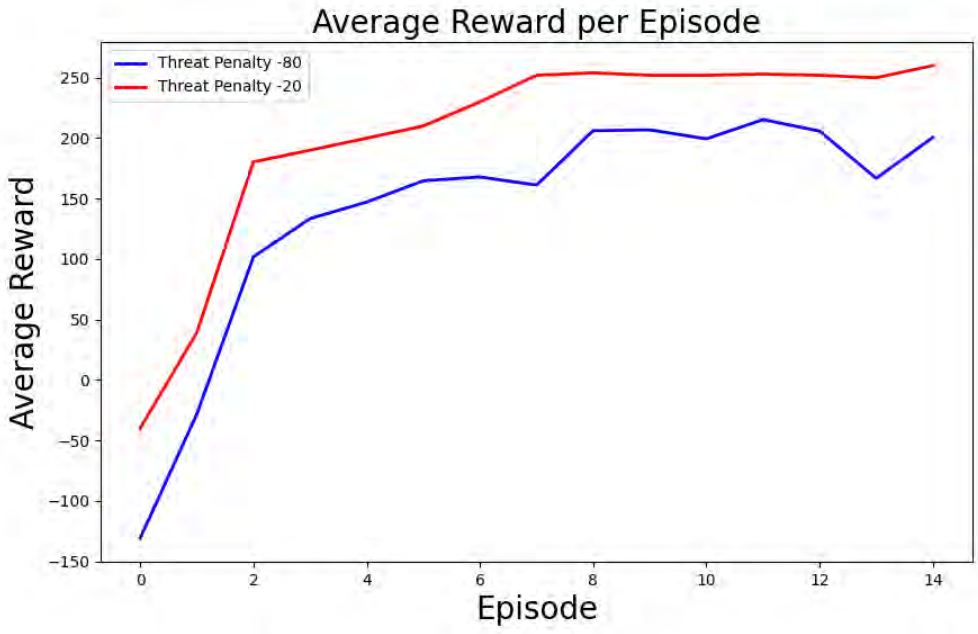

Figure. 20 Reward plot for A3C agent with different penalty, moving threat.

Therefore, the final reward for agents were set as +200 for finding the exit, -20 for encounter the threats, and -0.1 for every step moved.

Section 4: Results

To evaluate and summarize the findings of the evacuation under different settings, the results were classified and compared across three distinct scenarios: a single-agent static threat environment, a single-agent moving threat environment, and a multi-agent environment. In each scenario, evacuation simulations were conducted using two different algorithms, A3C and DQN, to compare their efficiencies. This comparison involved measuring the time taken for each successful evacuation in seconds and

analyzing the learning process by evaluating the cumulative rewards per episode. These metrics helped assess the performance of the two algorithms under identical environmental conditions and settings.

Static Threat Environment

Efficiency was measured in terms of the average time in seconds taken for the single agent to evacuate from the static threat environment, Figure 15 shows a distinct advantage of A3C over DQN in the static threat environment. In the graph, the x-axis represents the number of episodes, and the y-axis shows the average evacuation time in seconds. As shown in Figure 21, A3C (in blue) consistently demonstrated a faster evacuation time across episodes compared to DQN (in red). The average evacuation time for A3C was approximately 22.32 seconds, and for DQN, it was about 39.75 seconds. This result suggests that A3C is approximately 43.86% faster than DQN in terms of the average time taken for agent evacuation. This percentage reflects the significant efficiency advantage of A3C over DQN in this scenario. The better efficiency of A3C compared to DQN may be attributed to the asynchronous learning process. In this scenario, the agent using the DQN algorithm learned and updated its strategy after completing each episode. In contrast, the A3C algorithm leveraged multiple parallel threads to learn and update strategies simultaneously, resulting in significantly faster exploration of the environment and policy learning compared to DQN. From the learning curve aspect, A3C and DQN also indicate different numbers of episodes required for the agents to reach a stable evacuation strategy. This comprehensive analysis was designed to assess how each algorithm adapts and optimizes its strategy in response to different types of challenges.

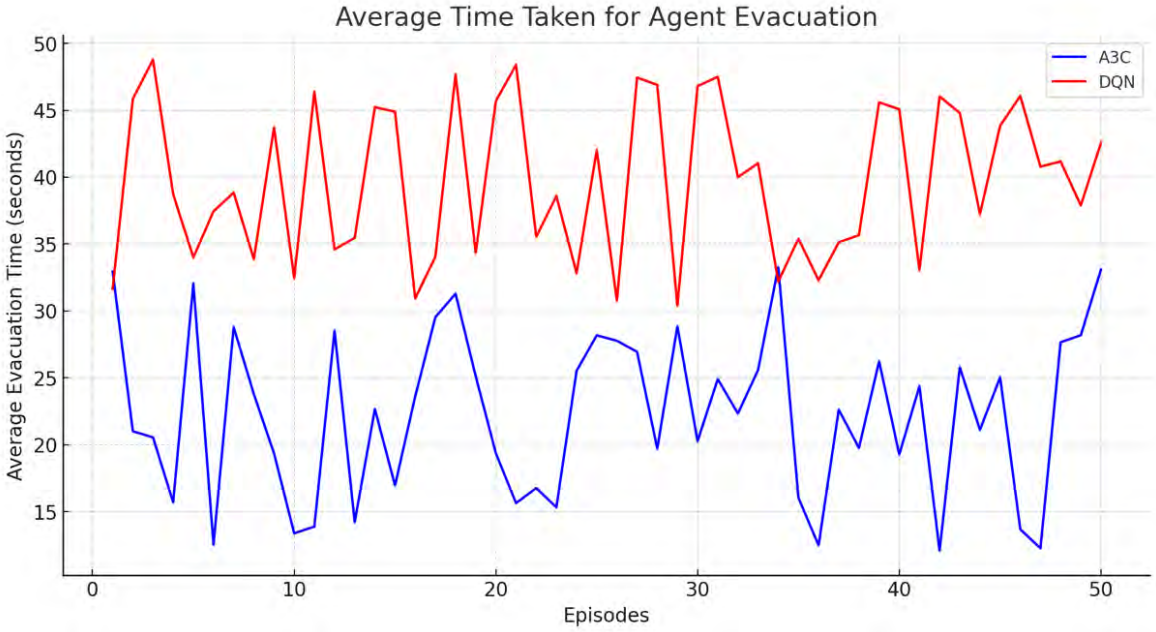

Figure 21. The Average Time Taken for Agent Evacuation Static Threat Environment

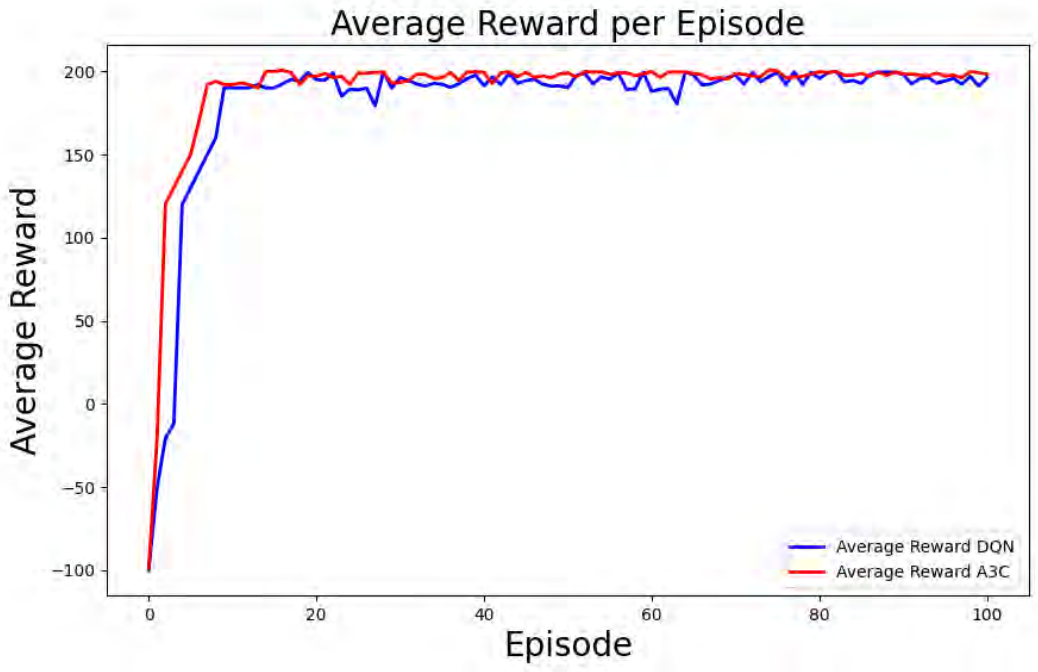

Figure 22. The Average Reward Comparison in Single Agent Static Environment

For the single agent static threat environment, the results indicate that both A3C and DQN displayed a similar pattern in the learning curve. The static nature of the threat allowed both algorithms to quickly learn and devise effective strategies. However, A3C continued to maintain a slight edge in terms of quicker strategy optimization, as indicated by its steeper learning curve in the initial episodes. A stable evacuation strategy was reached at around 100 to 200 episodes for A3C algorithm, while for DQN, the stable evacuation strategy was reached at around 200 to 300 episodes in this single agent static threat environment (Figure 22). This may be due to DQN's inherent learning mechanism, which might require more iterations to effectively capture and respond to the environmental parameters in this scenario.

Moving Threat Environment

In the single agent moving threat environment, the learning curves show a more notable difference in the efficiency and how each algorithm deal with the dynamic threats in the airport. A3C's response was swift and indicated a rapid decrease in evacuation time as the agent started to learn and navigate to avoid the moving threat. Meanwhile, DQN took longer time to adapt and form a stable evacuation strategy compared to the swift learning process of A3C as seen in the gradual slope of the learning curve (Figure 23).

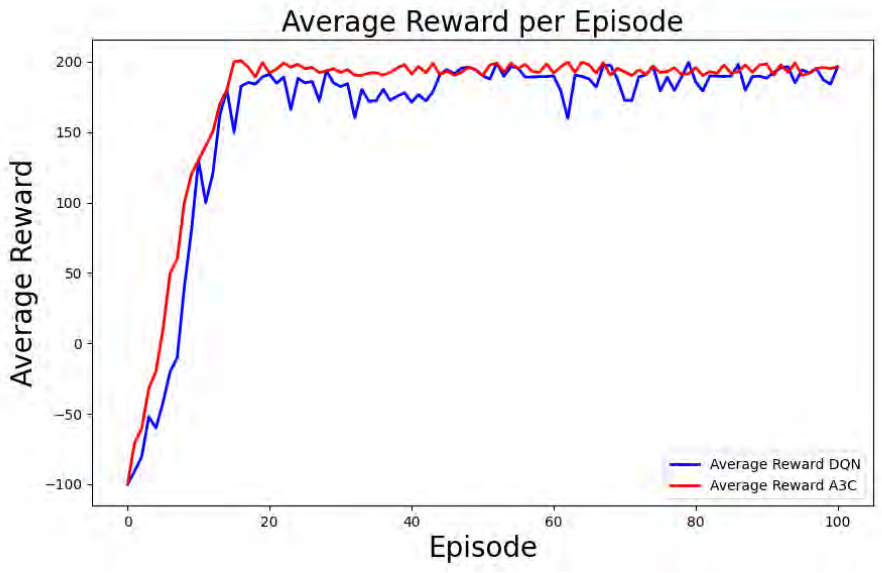

Figure 23. The Average Reward Comparison in Single Agent Moving Environment

Compared to the learning curve of static threat environment setting, the results generated in moving environment are more volatile, while the learning curves in static environments exhibited a relatively smooth progression towards stability. In the moving threat environment, both A3C and DQN displayed learning curves with significant fluctuations. The results were characterized by periodic fluctuations in the algorithms' learning curves across episodes, reflecting a continuous process of adaptation and re-adaptation to the dynamic nature of the moving threat, modeled as fire that expands and spreads in a specific pattern. These fluctuations highlight the A3C algorithm’s ability to support ongoing learning, enabling the agent to quickly adjust to evolving environmental conditions, including avoiding the advancing fire. Moreover, the observed variations in performance align with the core principles of the A3C algorithm: its asynchronous, multi-threaded learning structure allows for rapid response and adaptation but may also introduce temporary performance dips as new strategies are developed and integrated.

This trade-off demonstrates A3C's strength in navigating complex, unpredictable environments through continual policy refinement.

Multi-Agent Environment

In the multi-agent environment, only static threats were simulated, while moving threats did not achieve convergence. This limitation stems largely from the rule-based avoidance behavior implemented for multi-agent interactions. Consequently, the simulation may not fully capture realistic scenarios involving multiple agents and dynamic threats within a shared space. Several factors contribute to this challenge. Introducing multiple agents alongside moving threats significantly increases the complexity of the environment. In such settings, agents must learn not only to avoid continuously enlarging and advancing threats but also to navigate around one another. This requires a more sophisticated understanding of the environment, potentially necessitating longer training times and more advanced neural network architectures to achieve stable policy convergence.

The implementation of collision avoidance policies among multiple agents adds an additional layer of complexity. As the number of agents increases, so does the unpredictability of emergent behaviors. For instance, agents may begin clustering in perceived safe zones, an unintended strategy that may not be viable in real-world evacuation scenarios. Effective evacuation in such environments requires agents to learn cooperative behaviors, such as avoiding mutual obstruction while simultaneously evading dynamic threats. This interplay between cooperation and individual survival makes the learning process far more complex and poses a significant challenge for current reinforcement learning models.

In scenarios involving static threats, the asynchronous nature of the A3C algorithm demonstrated clear advantages, particularly in multi-agent environments. A3C allows multiple agents to learn simultaneously and synergistically, leading to a more efficient and accelerated learning process, as reflected in the steeper learning curve. While DQN is capable of learning cooperative strategies, it exhibited slower convergence in comparison, highlighting its limitations in handling the parallel learning tasks required for multi-agent evacuation scenarios.

A3C's strength in multi-agent environments stems from its asynchronous learning mechanism, which enables agents to independently explore and interact with the environment while collectively contributing to the global policy update. This parallelized approach not only speeds up the learning process but also facilitates the faster development of effective cooperative strategies among agents. Experimental results, as illustrated in Figure 24, confirm that A3C enables more rapid and efficient coordination, making it particularly well-suited for complex evacuation tasks involving multiple agents.

Conversely, while DQN can develop cooperative strategies, its inherently sequential and centralized learning approach poses challenges in multi-agent contexts. It struggles to manage simultaneous learning from diverse agent experiences, resulting in slower convergence toward optimal behaviors. This limitation reduces DQN’s ability to quickly adapt and integrate strategies in dynamic, multi-agent environments, where rapid coordination and mutual adaptation are critical for successful evacuation outcomes.

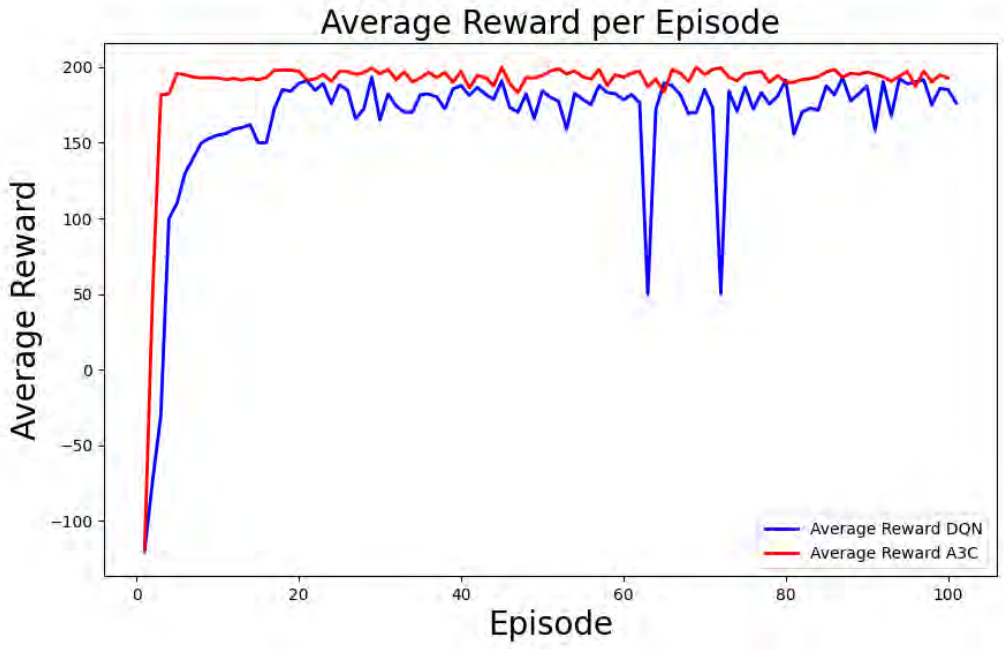

Figure 24. The Average Reward Comparison in the Multi-Agent Static Environment

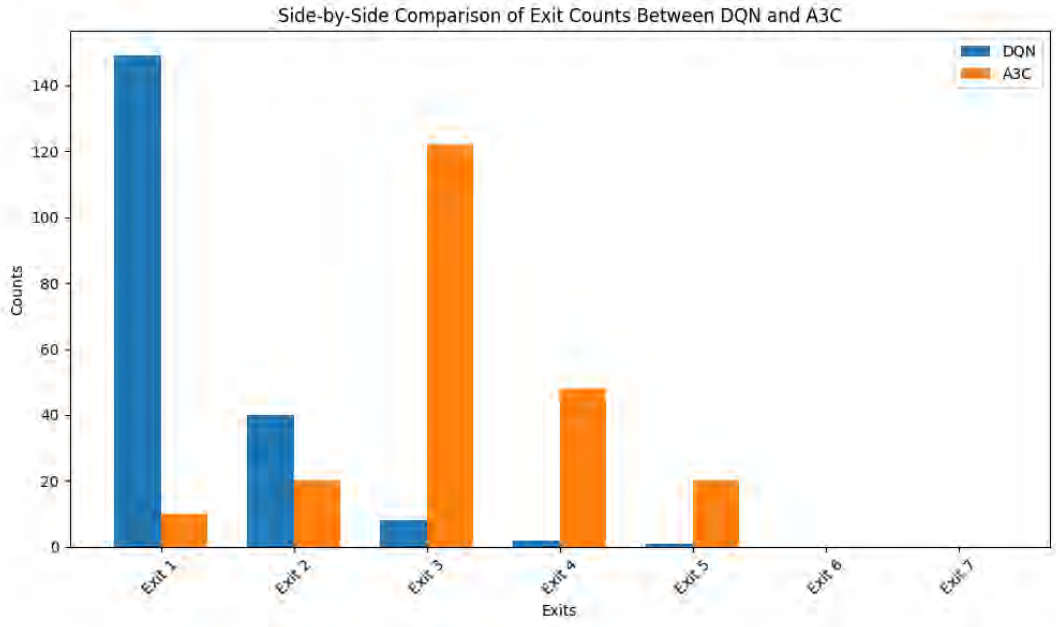

Figure 25 Comparison of Exit Count between DQN and A3C

A closer examination of exit selection reveals that the A3C algorithm identifies more optimal evacuation paths compared to DQN. As shown in Figure 25, A3C produces a more balanced distribution of exit choices, reflecting its ability to explore the environment more comprehensively. This thorough exploration enables agents to develop evacuation strategies that are not only more efficient but also better aligned with their initial positions, resulting in shorter travel distances and reduced evacuation times.

This advantage stems from A3C's asynchronous and concurrent learning capabilities, which allow agents to rapidly adapt and coordinate in complex, dynamic environments. Such adaptability is particularly valuable in scenarios like collaborative robotics and multi-agent simulations, where efficient strategy development and coordination are essential. Moreover, A3C's responsiveness to environmental changes highlights its potential for unpredictable, real-world applications, such as airport evacuations, where multiple agents (e.g., passengers) must navigate shifting threats and evolving conditions. Its ability to incorporate new information and adjust strategies in real time makes A3C a robust solution for environments requiring flexibility and rapid decision-making.

In summary, a significant performance gap was observed between A3C and DQN across different environments, including static threats, moving threats, and multi-agent settings. In all cases, A3C consistently demonstrated a more robust and accelerated learning process, outperforming DQN in identifying optimal evacuation routes and minimizing evacuation time. These outcomes are largely attributed to A3C's capacity for handling asynchronous updates and adapting more effectively to environmental complexity. While DQN performs adequately in simpler scenarios, such as single-agent

environments with static threats, it falls short in situations demanding real-time adaptation and coordinated behavior among multiple agents, capabilities in which A3C excels.

Section 5: Conclusion & Discussion

This study addressed the gap identified in the literature concerning the practical application of advanced computational models, specifically the A3C algorithm, in simulating realistic evacuation scenarios. Comparative analysis of the A3C and DQN models in simulating airport evacuations revealed that A3C outperforms DQN in several key areas: 1. Efficiency in Evacuation in terms of speed and correctness, 2. Adaptability to Dynamic Environments, 3. Robustness in Learning and Strategy Adaptation. Notably, A3C demonstrated superior adaptability and efficiency in managing dynamic threats and high-density environments, which are characteristic of real-life aviation emergencies. A3C's proficiency in adapting to changing conditions was particularly effective in scenarios with moving threats, highlighting its practical advantage in unpredictable situations. The A3C model also enabled faster evacuations and handled high-density crowds more effectively, minimizing bottlenecks and promoting smoother evacuation flows. Additionally, it demonstrated a robust learning curve and efficiently adapted its strategies in response to environmental changes, an area where the DQN model proved less effective.

The findings from this comparative analysis have profound implications for transportation safety and policymaking. The results of this study suggests that A3C is generally more versatile and efficient in various environmental settings, especially in scenarios that require quick adaptation and multi-agent learning. This study emphasizes

the significance of choosing an appropriate algorithm for specific environmental challenges. For instance, an airport evacuation represents a complex and constantly evolving scenario. A3C is a particularly strong algorithm for such dynamic real-world applications that require quick decision-making ability and adaptability due to its emerging speed, accuracy, and efficiency.

By illustrating A3C's effectiveness in managing dynamic threats and facilitating smooth evacuation flows, this study advocates for the integration of advanced computational models into the emergency response strategies of airports and other critical infrastructure. For policymakers and emergency planners, adopting A3C-based simulations can lead to more informed decision-making, enabling the design of evacuation protocols that are both safer and more efficient. Furthermore, these results can guide the development of policies that prioritize investments in technology-driven safety improvements, ensuring that transportation systems are resilient against a wide range of emergency scenarios. Utilizing the rapid adaptability of machine learning algorithms like A3C, the modern transportation system can dynamically optimize the shortest and safest evacuation routes in real time when emergency situations happen. This level of optimization can also expand to large-scale evacuations due to natural disasters or urban crises. A3C’s ability to adapt quickly to changing environmental factors such as traffic congestion, road closures, or hazardous conditions allows agents (passengers, pedestrians, vehicles etc.) to reroute efficiently for a safer and less congested path. Algorithms such as A3C and DQN can be effectively applied in the transportation industry for simulation and training purposes, offering a cost- and time-efficient approach to evaluating optimal evacuation plans under various emergency scenarios. These

simulations not only support the development of more robust evacuation strategies but also serve as valuable training tools for industry personnel, ultimately enhancing organizational preparedness and response effectiveness during real-world emergencies. Moreover, A3C's capabilities extend beyond evacuation planning. Its ability to manage multiple agents and process large volumes of data in real time makes it well-suited for implementation in intelligent traffic systems. In emergency situations, A3C can be used to dynamically control traffic flow by adjusting traffic signals, lane assignments, and routing strategies, thereby facilitating smoother evacuations and more efficient emergency responses.

However, certain limitations remain in both the algorithms and the scope of this research. One notable limitation of the A3C algorithm is its significant computational demand, particularly when applied to complex, multi-agent scenarios. This high resource requirement can restrict its applicability in environments with limited computational capacity or may necessitate substantial investment in advanced hardware and infrastructure to ensure efficient performance. Additionally, the simulations may not fully capture the unpredictability and variability of human behavior in real-world emergency situations, which could impact the accuracy and applicability of the findings. Furthermore, the reliance on specific simulation parameters introduces limitations regarding the generalizability of the results across different types of emergencies or transportation settings, as variations in context, infrastructure, and population dynamics may produce different outcomes.

For future research, a more systematic exploration of reward and penalty values is recommended to determine optimal configurations for the simulation environment. As

observed in the results, these values significantly influence agent behavior, and identifying the most effective settings could enhance the overall performance of evacuation strategies.

Integrating the social force model with existing evacuation algorithms could also offer substantial improvements. The social force model accounts for individuals' anticipatory adjustments in response to social interactions, not just physical obstacles, which would allow for more realistic simulation of crowd dynamics. This integration could lead to more accurate representations of human movement, thereby optimizing evacuation strategies and improving safety in complex environments such as airports. Furthermore, incorporating demographic and physical attributes, such as average body dimensions and movement speeds of men, women, and children, would increase the realism of the model. Including variations in evacuee speed based on age or physical condition would better reflect real-world conditions and enhance the model's ability to simulate diverse populations under stress.

To further improve accuracy and efficiency, it is also important to incorporate more complex and dynamic elements into the simulation. This includes modeling diverse human behavior patterns, accounting for intricate architectural features, and simulating unpredictable environmental conditions. By doing so, the model would more closely mimic real-world emergency scenarios and become better suited for supporting complex, real-time decision-making processes. With these enhancements, the simulation framework could be expanded to support a broader range of emergency planning applications, ultimately contributing to more effective evacuation procedures in high-stakes, real-world settings.

References

Alexander, D. E. (2013). Emergency, definition of. In K. B. Penuel, M. Statler, & R. Hagen (Eds.), *Encyclopedia of crisis management* (pp. 324–325). SAGE Publications.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *arXiv*. https://doi.org/10.5281/zenodo.16014

Bureau of Transportation Statistics. (2021). *On-time performance: Causes of delay*. U.S. Department of Transportation. https://www.transtats.bts.gov/ot_delay/ot_delaycause1

Chen, J., Liu, D., Namilae, S., Sang-A, L., Thropp, J. E., & Seong, Y. (2019). Effects of exit doors and number of passengers on airport evacuation efficiency using agent-based simulation. *International Journal of Aviation, Aeronautics, and Aerospace, 6*(5), Article 3. https://doi.org/10.15394/ijaaa.2019.1418

Cheng, L., Reddy, V., Fookes, C., & Yarlagadda, P. K. D. V. (2014). Impact of passenger group dynamics on an airport evacuation process using an agent-based model. In *2014 International Conference on Computational Science and Computational Intelligence* (pp. 161–167). IEEE. https://doi.org/10.1109/CSCI.2014.111

Christensen, K., & Sasaki, Y. (2008). Agent-based emergency evacuation simulation with individuals with disabilities in the population. *Journal of Artificial Societies and Social Simulation, 11*(3), 9. http://jasss.soc.surrey.ac.uk/11/3/9.html

Ding, G., Li, X., Shi, X., & Yu, P. (2021). Data transmission evaluation and allocation mechanism of the optimal routing path: An asynchronous advantage actor-critic

(A3C) approach. *Wireless Communications and Mobile Computing, 2021*, 1–21. https://doi.org/10.1155/2021/6685722

Gao, H., Xu, J., Li, S., & Xu, L. (2019). Forecast of passenger flow under the interruption of urban rail transit operation. In *Proceedings of the 4th International Conference on Electrical and Information Technologies for Rail Transportation (EITRT)* (pp. 283–291). Springer. https://doi.org/10.1007/978-981-13-8323-6_27

Gosavi, A. A. (2004). Reinforcement learning algorithm based on policy iteration for average reward: Empirical results with yield management and convergence analysis. *Machine Learning, 55*, 5–29. https://doi.org/10.1023/B:MACH.0000019802.64038.6c

Gota, Puscasiu, A., Fanca, A., Valean, H., & Miclea, L. (2020). Threat objects detection in airport using machine learning. In *2020 21st International Carpathian Control Conference (ICCC)* (pp. 1–6). IEEE. https://doi.org/10.1109/ICCC49264.2020.9257293

Gu, Z., Jia, Z., & Choset, H. (2019). Adversary A3C for robust reinforcement learning. *arXiv*. https://arxiv.org/abs/1912.00330

Helbing, D. (1993). Traffic and related self-driven many-particle systems. *Physica A: Statistical Mechanics and its Applications, 196*(4), 546–573.

Helbing, D., & Molnar, P. (1995). Social force model for pedestrian dynamics. *Physical Review E, 51*(5), 4282–4286.

Hradecky, S. (2016, October 28). Accident: American B763 at Chicago on Oct 28th 2016, rejected takeoff, fire at right hand wing due to uncontained engine failure. *The Aviation Herald.* http://avherald.com/h?article=49ffa115&opt=0

Hradecky, S. (2017, July 3). Incident: Skywest CRJ7 at Denver on Jul 2nd 2017, engine fire on landing. *The Aviation Herald.* http://avherald.com/h?article=4ab23fdb&opt=0

Hradecky, S. (2017, July 11). Incident: Delta A320 near Daytona Beach on Jul 10th 2017, hail strike. *The Aviation Herald.* http://avherald.com/h?article=4ab7c100&opt=0

Kim, B., & Pineau, J. (2015). Socially adaptive path planning in human environments using inverse reinforcement learning. *International Journal of Social Robotics, 8*(1), 51–66. https://doi.org/10.1007/s12369-015-0310-2

Lee, J., & Yoo, H. (2023). *Deep reinforcement learning processor design for mobile applications*. Springer. https://doi.org/10.1007/978-3-031-36793-9_1

Liu, R., Jiang, D., & Shi, L. (2016). Agent-based simulation of alternative classroom evacuation scenarios. *Frontiers of Architectural Research, 5*(1), 111–125.

Ma, J., Zeng, X., Xue, X., & Deng, R. (2022). Metro emergency passenger flow prediction on transfer learning and LSTM model. *Applied Sciences, 12*(3), 1644. https://doi.org/10.3390/app12031644

Martinez-Gil, F., Lozano, M., & Fernández, F. (2011). Multi-agent reinforcement learning for simulating pedestrian navigation. In *International Workshop on Adaptive and Learning Agents* (pp. 54–69).

Martin, T. (2023). Fire spread model. *MATLAB Central File Exchange*. https://www.mathworks.com/matlabcentral/fileexchange/74499-fire-spread-model

Mitchell, T. (1997). *Machine learning*. McGraw-Hill.

Miyoshi, T., Nakayasu, H., Ueno, Y., & Patterson, P. (2012). An emergency aircraft evacuation simulation considering passenger emotions. *Computers & Industrial Engineering, 62*(3), 746–754. https://doi.org/10.1016/j.cie.2011.11.012

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning* (pp. 1928–1937). PMLR.

Papoudakis, G., Christianos, F., Schäfer, L., & Albrecht, S. V. (2020). Comparative evaluation of multi-agent deep reinforcement learning algorithms. *arXiv*. https://arxiv.org/abs/2006.07869

Rossier, R. (n.d.). Emergency landings. *AOPA*. https://www.aopa.org/training-and-safety/students/flighttestprep/skills/emergency-landings

Sandoval, E. (2017, July 10). Cracked windshield forces Haiti-bound Delta jet to land in Daytona. *ClickOrlando.* https://www.clickorlando.com/news/cracked-windshield-forces-haiti-bound-delta-flight-to-land-in-daytona

Skanda Vaidyanath, Georgila, K., & Traum, D. (2020). Using reinforcement learning to manage communications between humans and artificial agents in an evacuation scenario. In *Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference*. https://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS20/paper/view/18469/17622

StepsApp. (n.d.). Did you know about the step length? https://steps.app/en/blog/stepcounting/did-you-know-about-the-step-length

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.

Tian, K., & Jiang, S. (2018). Reinforcement learning for safe evacuation time of fire in Hong Kong-Zhuhai-Macau immersed tube tunnel. *Systems Science & Control Engineering, 6*(2), 45–56.

U.S. Department of Transportation, Federal Aviation Administration. (2004). *Federal Aviation Rule (FAR) 139.325: Airport emergency plan*.

U.S. Department of Transportation, Federal Aviation Administration. (2009). *Advisory Circular (AC) 150/5200-31C: Airport emergency plan*.

Villamizar, H. (2022, June 22). How airport emergency response works. *Airways Magazine.* https://airwaysmag.com/airport-emergency-response/

Vorst, R. (2010). Evacuation models and disaster psychology. *Procedia Engineering, 3*, 15–21. https://doi.org/10.1016/j.proeng.2010.07.004

Wang, Q., Liu, H., Gao, K., & Zhang, L. (2019). Improved multi-agent reinforcement learning for path planning-based crowd simulation. *IEEE Access, 7*, 73841–73855.

Yao, W., Zhang, G., Lu, D., & Liu, H. (2019). Data-driven crowd evacuation: A reinforcement learning method. *Neurocomputing, 366*, 314–327. https://doi.org/10.1016/j.neucom.2019.08.021

Yang, Y., Zhang, K., Liu, D., & Song, H. (2020). Autonomous UAV navigation in dynamic environments with double deep Q-networks. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)* (pp. 1–7). IEEE. https://doi.org/10.1109/DASC50938.2020.9256455

Zarraonandia, T., Vargas, M. R. R., Díaz, P., & Aedo, I. (2009). A virtual environment for learning airport emergency management protocols. In *International*

Conference on Human-Computer Interaction (pp. 228–235). Springer. https://doi.org/10.1007/978-3-642-02774-1_25

Zhang, C., Chai, Z., & Lykotrafitis, G. (2021). Deep reinforcement learning with a particle dynamics environment applied to emergency evacuation of a room with obstacles. *Physica A, 571,* 125845. https://doi.org/10.1016/j.physa.2021.125845

Zhang, S., & Guo, Y. (2015). Distributed multi-robot evacuation incorporating human behavior. *Asian Journal of Control, 17*(1), 34–44.

Zhang, G., Zhu, G., Yuan, G., & Wang, Y. (2016). Quantitative risk assessment methods of evacuation safety for collapse of large steel structure gymnasium caused by localized fire. *Safety Science, 87,* 234–242.