A Reinforcement Learning-based Scheduler for Minimizing Casualties of a Military Drone Swarm

Heng Jin

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science and Applications

June 10, 2022

Blacksburg, Virginia

Keywords: Drone swarm, Casualty, Scheduling, Age of Information, Reinforcement learning

A Reinforcement Learning-based Scheduler for Minimizing Casualties of a Military Drone Swarm

Heng Jin

(ABSTRACT)

In this thesis, we consider a swarm of military drones flying over an unfriendly territory, where a drone can be shot down by an enemy with an age-based risk probability. We study the problem of scheduling surveillance image transmissions among the drones with the objective of minimizing the overall casualty. We present Hector, a reinforcement learning-based scheduling algorithm. Specifically, Hector only uses the age of each detected target, a piece of locally available information at each drone, as an input to a neural network to make scheduling decisions. Extensive simulations show that Hector significantly reduces casualties than a baseline round-robin algorithm. Further, Hector can offer comparable performance to a high-performing greedy scheduler, which assumes complete knowledge of global information.

A Reinforcement Learning-based Scheduler for Minimizing Casualties of a Military Drone Swarm

Heng Jin

(GENERAL AUDIENCE ABSTRACT)

Drones have been successfully deployed by the military. The advancement of machine learning further empowers drones to automatically identify, recognize, and even eliminate adversary targets on the battlefield. However, to minimize unnecessary casualties to civilians, it is important to introduce additional checks and control from the control center before lethal force is authorized. Thus, the communication between drones and the control center becomes critical. In this thesis, we study the problem of communication between a military drone swarm and the control center when drones are flying over unfriendly territory where drones can be shot down by enemies. We present Hector, an algorithm based on machine learning, to minimize the overall casualty of drones by scheduling data transmission. Extensive simulations show that Hector significantly reduces casualties than traditional algorithms.

Acknowledgments

First, I would like to express my heartfelt gratitude to my advisor, Prof. Tom Hou, the Bradley Distinguished Professor of ECE at Virginia Tech. I was originally inquiring a GRA position from Prof. Hou. I was fortunate to be admitted to Prof. Hou's group to pursue a M.S. degree. Throughout my M.S. study, Prof. Hou has provided me with valuable support and guidance. He spent numerous hours with me to discuss the demo that I was assigned to work on, and later, my thesis and research paper. He went through all technical details of my work and helped me with writing a paper for submission. It would be impossible for me to finish my study without his guidance. Prof. Hou taught me how to describe ideas in a approach that average people can easily comprehend. Collectively, the knowledge and skills that he passed on to me are the most precious thing I learned at Virginia Tech.

My sincere gratitude extends to Dr. Qingyu Liu. Dr. Liu has been offering guidance of my work since the very beginning. He has been more than a scholarly mentor, but also become a trustworthy friend in the process. He personally offered me car ride to school when commuting was a challenge to me, even though it would take a detour for him in the process. Dr. Liu went through my thesis word by word to improve the writing quality. I am greatly indebted to his help and friendship.

I would like to express my gratitude to Prof. Wenjing Lou. I thank her for serving on my thesis committee and being prompt and supportive in the process. She has offered me valuable feedback and suggestions during the demo development phase and on my thesis.

Finally, I would like to express my gratitude to my family and friends for their love, encouragement, and understanding throughout my study abroad.

Funding Acknowledgments

This research was supported in part by ONR under MURI Grant N00014-19-1-2621, Virginia Commonwealth Cyber Initiative (CCI), and Virginia Tech Institute for Critical Technology and Applied Science (ICTAS).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Drones have been successfully deployed by the military and are carrying out missions without incurring the risks associated with human warfighters [4, 9, 21, 23, 26, 28]. The advancement of machine learning further empowers drones to automatically identify, recognize, and even eliminate adversary targets on the battlefield[8, 14, 16]. But drones have their limitations. To minimize unnecessary casualties to civilians and other targets from potential machine error, it is important to introduce additional checks and control before a lethal force is exercised.

In this paper, we consider a swarm of N military drones flying over an unfriendly territory with hostile enemies (see Fig. 1.1 and our video demo in [15]). When a drone flies over the unfriendly territory, it is exposed to enemies' fire and becomes vulnerable. The risk probability for a drone is a complex function that depends on both the number of enemies and the amount of time that the drone has been in each enemy's territory, i.e., the age of this drone at each enemy's side.

When a drone identifies a potentially target, it will capture its image and try to convey the image to a base station (BS) for a final determination before exercising a lethal action. All drones share the same communication channel to transmit images, and the uplink wireless channel is the main bottleneck and the root of information latency. Once the BS receives an image, it makes a determination on the nature of the target and issue a final confirmation of kill/no-kill to the respective drone via a downlink control channel. With potentially many

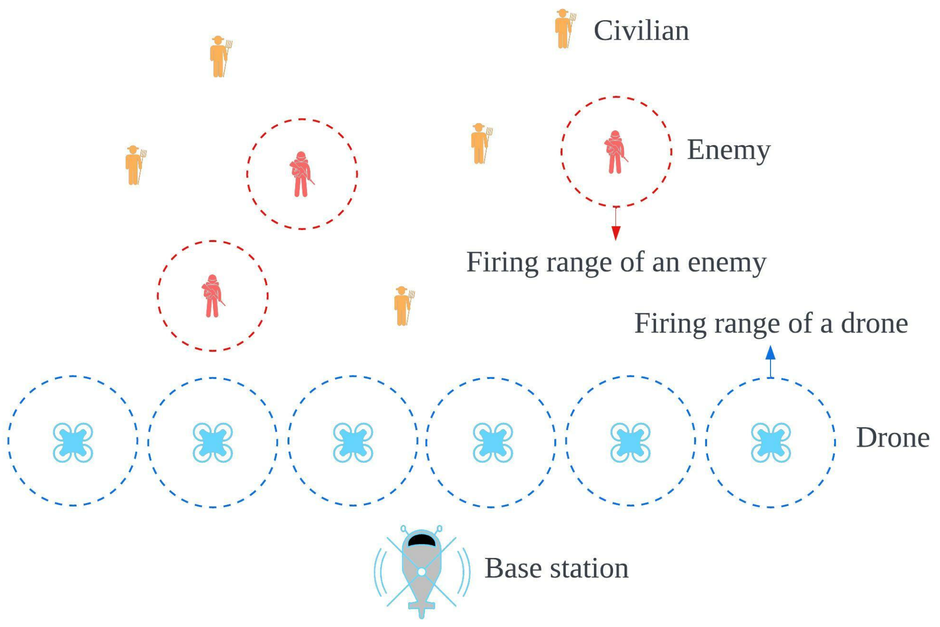


Figure 1.1: A swarm of drones flying over an unfriendly territory containing either hostile enemies or unarmed civilians.

drones attempting to transmit different target imagery to the BS in the uplink, a scheduler is needed to coordinate the order of image transmissions. This paper investigate this scheduling problem, with the goal of minimizing overall casualty of the drones for the mission. The main contributions of this paper are summarized as follows:

- For a drone swarm flying over an unfriendly territory, we characterize the probability of a drone being shot down (risk probability) by an age-based function. We formulate a scheduling problem for transmitting surveillance images among the drones to the BS with an objective of minimizing the overall casualty. For this problem, we develop a greedy scheduling algorithm. For each drone, the greedy algorithm first calculates the expected number of drones to be shot down if the drone is selected for transmission. Then the greedy algorithm compares the expected number of casualties among all drones and select the drone corresponding to the lowest casualty for transmission.

- A fundamental limitation of the greedy algorithm is that it requires global information, including those only available at enemies' side. As a result, this greedy algorithm can only be used as a benchmark, rather than the actual scheduler that can be deployed in the field. To address the limitation of the greedy algorithm, we propose Hector—a Reinforcement Learning (RL) based online scheduling algorithm that only uses information locally available at the drones. It uses the age of each detected enemy at the drone's side as an input to a neural network that is trained by Proximal Policy Optimization (PPO). The output of the neural network gives the probabilities for drones in the swarm, and the drone with the highest probability will be selected for transmission.

- We conduct extensive simulation experiments to evaluate the performance of Hector. Our results show that the performance of Hector is similar to the greedy algorithm (despite that Hector only uses local information while the greedy algorithm requires global information), and both Hector and the greedy algorithm significantly outperform a baseline round-robin algorithm.

1.1 System Model and Problem Description

1.1.1 A Military Drone Swarm in the Battlefield

Consider a scenario where a swarm of military drones forming an array and flying in the same direction (e.g., upward) over an unfriendly territory (see Fig. 1.1). From the drones' side, each drone captures images of potential targets from its view. Although the drone is able to identify potential target and take images, we require that the drone must obtain explicit confirmation from the central command and control center (the BS) before it exercises any lethal action. This confirmation is especially important for two reasons. First, from humanity

Table 1.1: Important notations.

Notation	Definition
R_d	Radius of drone d's visual area
R_e	Radius of enemy e's firing area
$A_{d(e)}(t)$	Age of enemy d since it enters drone d's visual area
$D_{e(d)}(t)$	Age of drone d since it enters enemy e's firing area
δ_k	Starting time for transmitting the k-th image received by the BS
$\boldsymbol{S}(\delta_k)$	Scheduling decision at time δ_k
$S_i(\delta_k)$	i-th element of $\boldsymbol{S}(\delta_k)$
$\alpha_{e(d)}(t)$	Risk probability for drone d to be destroyed by enemy e at time t
$p_d(t)$	Risk probability for drone d to be destroyed at time t

perspective, loss of civilian lives must be avoided at all possibility, even though doing so may put the drone at a higher risk of being shot down. Second, the processing and storage capability at a drone is very limited, and thus, sophisticated target classification is better done at the central command and control center, which is equipped with much more powerful processing and storage capacity.

In Fig. 1.1, we assume the central command and control BS resides in a helicopter, which follows behind the array of drones in the upward direction. When the BS receives an image from a drone, it will process the image to make a determination of whether the first target in the image is a hostile enemy or civilian. If the target in the image is indeed an enemy and should be eliminated, the BS sends its kill authorization to the drone via a control channel. Otherwise, no action will be taken.

Given that the uplink wireless channel (from the drones to the BS) is shared, different drones cannot transmit their images simultaneously and a scheduler is needed. Upon the completion of an image transmission from a drone, the scheduler needs to decide which drone will be selected for the next image transmission.

On the enemy's side, a drone is exposed and considered to be vulnerable (to be shot down) when it is within the enemy's effective firing range. Once a drone enters an enemy's firing range, its risk probability (of being shot down) increases with time as long as it remains in this enemy's firing range. Note that we model the risk (vulnerability) of a drone in a probability sense (instead of deterministically) for a variety of reasons. For example, an enemy may choose not to fire at the drone if it wishes to conceal its position. Another example is that the enemy may be currently devoted to other battlefield activities before shifting its attention to the drone in the sky.

In this battlefield scenario, it is critical to have the drones to transmit their target images to the BS and receive kill confirmation in time to minimize their casualties. That is, we need to a scheduler that can coordinate the transmission of images from the drones so that the overall casualties of the drones are minimized.

1.2 System Model

1.2.1 Age of Information Definitions

For drone d, we assume its visual area is a circular and has a radius R_d. We also assume that there is no overlap of visual areas of different drones. Denote $t^E_{d(e)}$ and $t^L_{d(e)}$ as the time instances when enemy e enters and leaves the visual area of drone d, respectively, where we use $d(e)$ to indicate that enemy e is in the visual area of drone d. Then the "age" of enemy e since it enters drone d's visual area, denoted as $A_{d(e)}(t)$, can be defined as:

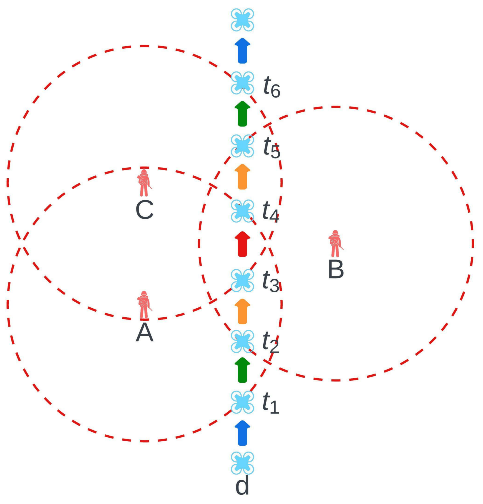
$$A_{d(e)}(t) = \begin{cases} t - t^E_{d(e)}, & \text{if } t^E_{d(e)} \le t \le t^L_{d(e)}; \\ -\infty, & \text{otherwise.} \end{cases} \tag{1.1}$$

where we use $-\infty$ to indicate that age is undefined when the enemy falls outside a drone's visual area.

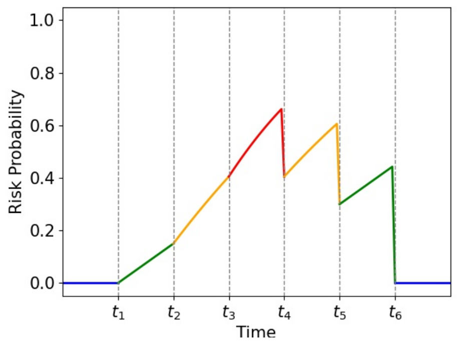
For enemy e, we assume its firing area is also circular and has a radius R_e. Denote $\tau^E_{e(d)}$ and $\tau^L_{e(d)}$ as the time instances when drone d enters and leaves the firing area of enemy e, respectively, where we use $e(d)$ to indicate that drone d is in the firing area of enemy e. Then the age of drone d since it enters enemy's e's firing area, denoted as $D_{e(d)}(t)$, can be defined as follows:

$$D_{e(d)}(t) = \begin{cases} t - \tau^E_{e(d)}, & \text{if } \tau^E_{e(d)} \leq \tau \leq \tau^L_{e(d)}; \\ -\infty, & \text{otherwise.} \end{cases} \tag{1.2}$$

Note that our definition for age (i.e., $A_{d(e)}(t)$ and $D_{e(d)}(t)$) is the same as the Age of Information (AoI) definition by Kaul *et al.* [17, 18], which is defined to be the elapsed time between the present and the generation time of source's information currently stored at the destination. A rather complete list of research papers on AoI can be found on a website maintained by Sun [27]. Although AoI has been considered for trajectory design of drones in drone-aided wireless networks where drones serve as relays for information transmission (see, e.g., in [2, 3, 5, 6, 11, 12, 13, 29, 30]), to date, we have not seen any research efforts on employing AoI as an input parameter to design a scheduler for a military operation involving a drone swarm.



(a) Drone d flies over multiple enemies' firing ranges.



(b) Evolution of $p_d(t)$ with time t.

Figure 1.2: An example of risk probability $p_d(t)$ for drone d as it flies over multiple enemies' firing ranges over time t.

1.2.2 Risk Probabilities

Denote $\alpha_{e(d)}(t)$ as the probability for enemy e to shoot down drone d at time t. We call $\alpha_{e(d)}(t)$ as the risk probability of drone d and define it as:

$$\alpha_{e(d)}(t) = \begin{cases} \min\left\{f\left(D_{e(d)}(t)\right), 1\right\}, & \text{if } \tau^E_{e(d)} \le t \le \tau^L_{e(d)}; \\ 0, & \text{otherwise,} \end{cases} \tag{1.3}$$

where $f(\cdot)$ is a non-decreasing function of $D_{e(d)}(t)$.

When drone d in within fire ranges of multiple enemies, its risk probability is a bit more complicated. Denote $p_d(t)$ as the risk probability for drone d to be destroyed at time t. Then we have

$$p_d(t) \;=\; 1 - \prod_e (1 - \alpha_{e(d)}(t)). \tag{1.4}$$

where $(1 - \alpha_{e(d)}(t))$ is the probability for enemy e not to shoot at drone d at time t and $\prod_e(1 - \alpha_{e(d)}(t))$ is the joint probability that all enemies in which drone d falls will not shoot at d.

Example 1.1. Consider the following linear function $f(\cdot)$ in (1.3) for all drones d's and all enemies e's:

$$f\left(D_{e(d)}(t)\right) = c \cdot D_{e(d)}(t), \tag{1.5}$$

and assume that $c = 0.001$. Fig. 1.2a shows that at time t_1 the drone enters the firing area of enemy A. Fig. 1.2b shows how $p_d(t)$ evolves with time t. As time moves on, at time t_2 the drone enters the firing area of enemy B. During $[t_2, t_3]$, the drone falls in the firing range of both enemies A and B. At time t_3 the drone enters the firing range of enemy C. During time $[t_3, t_4]$, the drone falls in the firing ranges of enemies A, B, and C. At time t_4, the drone leaves the firing area of enemy A; at t_5 the drone leaves the firing area of enemy B. Finally at time t_6 the drone leaves the firing area of enemy C. As shown in Fig. 1.2b, $p_d(t)$ is a complex function over time and is calculated based on (1.4). ■

In this paper, we assume that whenever a drone is shot down, the BS (helicopter) will immediately replenish a new drone to fill its vacant position. Also, when a drone receives a kill authorization from the BS, it will immediately eliminate the corresponding target with a probability of 1.

1.3 Uplink Image Transmission and Target Elimination

Denote δ_k $(k \in \mathbb{Z}^+)$ as the time of the k-th image transmission from the drones to the BS. Then at δ_k^-, the scheduler at the BS needs to decide which drone will be selected for transmitting its image at time δ_k. Since we have N drones lining in an array and at most one drone can transmit at δ_k, we can define an $N \times 1$ vector, $\boldsymbol{S}(\delta_k)$, as the scheduling decision at time δ_k, where the i-th element in $\boldsymbol{S}(\delta_k)$, denoted as $S_i(\delta_k) \in \{0, 1\}$, is 1 if the i-th drone is selected (scheduled) for transmission and 0 otherwise. Since at most one drone can transmit, we have:

$$\sum_i S_i(\delta_k) \ \leq \ 1, \ \text{for all } k \in \mathbb{Z}^+. \tag{1.6}$$

In this paper, we assume that a drone will send only one target in its image to the BS for confirmation, even though it may have already identified multiple potential targets. Denote $e_d(\delta_k)$ as the target that drone d has identified and will request a confirmation from the BS at time δ_k. Upon receiving this image for target $e_d(\delta_k)$ from drone d, the BS will make a determination on the nature of this target and issue kill/no-kill authorization. Such authorization will be sent to drone d in the control channel. For a kill authorization, drone d will eliminate the target immediately upon receiving this authorization.

1.4 Problem Statement and Technical Challenge

In our problem, we assume the swarm of military drones flies over an unfriendly territory via a line formation and it takes a fixed amount of time to complete the flight (under a given speed). So our goal is to find a scheduler so that the done casualties (i.e., the number of drones that are shot down) are minimized over this period.

There are a number of significant challenges with our problem. First, as illustrated in Example 1.1, the risk probability $p_d(t)$ for a drone d is highly complex and depends on both the enemies present along the drone's path as well as recent scheduling decisions. That is, if an enemy is eliminated by the drone, then it will no longer pose any risk to the drone in the future. Second, the scheduling algorithm that we need to design is an online algorithm, in the sense that it does not have any knowledge (enemy, scheduling) of the future. It is well known that it is impossible to design a provably optimal online scheduler, so we can only resort to good heuristics at best.

1.5 Solution Roadmap

To solve our scheduling problem, in Section 2, we first develop a greedy algorithm that assumes full knowledge of the following information at each scheduling time δ_k:

- $f(\cdot)$ and $D_{e(d)}(\delta_k)$ for all drone d and all enemy e.

Note that both of the above information is only available on the enemies' side and unknown to the drones. Therefore, the greedy algorithm is idealistic and can only serve as a benchmark.

To address the above fundamental limitation associated with the greedy algorithm, in Section 3 we design a Reinforcement Learning (RL)-based algorithm. This algorithm eliminates the requirement of knowledge for $D_{e(d)}(\delta_k)$ and $f(\cdot)$. Instead, it only assume knowledge of the following information at each scheduling time δ_k:

- $A_{d(e)}(\delta_k)$ at all drones d for all enemies e,

which is locally available at the drones and can be readily shared with the BS.

Chapter 2

A Greedy Scheduler Assuming Global Knowledge

2.1 Basic Idea

At each scheduling time δ_k, $k = 1, 2, \cdots$, the greedy algorithm needs to select which drone to transmit its image. Recall that if drone d is selected for image transmission, the potential target in the image can be confirmed by the BS immediately, which will in turn notify the drone to take lethal action against the target. Our greedy scheduler aims to achieve the smallest casualty after each image transmission. To do this, for each drone d, we will calculate the expected number of drones in the swarm to be shot down immediately after δ_k (i.e., δ_k^+) if drone d is selected for transmission at time δ_k. Then we will compare the expected number of casualties among all drones at time δ_k^+ and select the drone corresponding to the lowest casualty should it be selected for transmission.

2.2 Algorithm Details

Denote N as the number of drones flying in an array formation in Fig. 1.1. We define a conditional random variable $x(\delta_k^+|d) \in \{0, 1, \cdots, N\}$ as the number of drones to be destroyed at time δ_k^+ if (conditioned upon) drone d is scheduled for transmission at time δ_k. Recall that if d is scheduled for transmission at time δ_k, then upon receiving a kill confirmation from the BS at time δ_k^+, it will eliminates enemy target $e_d(\delta_k)$. The the expectation of $x(\delta_k^+|d)$ can be calculated as follows:

$$\begin{aligned}\mathbb{E}\left[x\left(\delta_k^+|d\right)\right] &= \sum_{i=1}^{N}\left(1\cdot p_i(\delta_k^+|d)+0\cdot(1-p_i(\delta_k^+|d))\right)\\ &= \sum_{i=1}^{N} p_i(\delta_k^+|d), \end{aligned} \tag{2.1}$$

where $p_i(\delta_k^+|d)$ is the probability for drone i to be destroyed under the condition that drone d is scheduled for transmission, i.e.,

$$p_i(\delta_k^+|d) = \begin{cases} 1-\prod_{e\neq e_d(\delta_k)}(1-\alpha_{e(i)}(\delta_k^+)), & \text{if } i=d;\\ 1-\prod_{e}(1-\alpha_{e(i)}(\delta_k^+)), & \text{otherwise.}\end{cases} \tag{2.2}$$

As mentioned earlier, the greedy algorithm schedule drone d corresponding to the smallest $\mathbb{E}\left[x\left(\delta_k^+|d\right)\right]$ for transmission at each time δ_k.

2.3 Limitations

There is a serious limitation with the greedy algorithm. In (2.2), we see that for each drone i, $p_i(\delta_k^+|d)$ is a function of $\alpha_{e(j)}(\delta_k^+)$ for all drone j's and enemies e's except for enemy $e_d(\delta_k)$. By definition (1.3), the calculation of $\alpha_{e(i)}(\delta_k)$ requires the knowledge of $f(\cdot)$ and $D_{e(i)}(\delta_k)$, which are only available on the enemies' side and unknown to the drones. That is, the greedy algorithm require global knowledge (including those only available at enemies' side) which is impossible to obtain in practice. As a result, this greedy algorithm can only serve as a performance benchmark.

Chapter 3

Hector: A RL-based Scheduler with Only Local Information

To address the problem with the greedy algorithm, we propose Hector[1]—a model-free RL-based scheduling algorithm that only relies on information locally available at the drones. Hector does not require knowledge of $f(\cdot)$ and $D_{e(d)}(\delta_k)$. Instead, it uses $A_{d(e)}(\delta_k)$ as measured at drone d for enemies e's as inputs to a neural network that is trained by Proximal Policy Optimization (PPO). The output of the neural network gives the probabilities for each drone, and the drone with the highest probability will be chosen to transmit its image.

3.1 Why Model-free RL

Before we consider machine learning-based approaches to our problem, it is important to have a clear understanding of the mathematical nature of our problem.

First, the objective function (i.e., the casualty of drones) cannot be modeled in closed form. This is due to the highly dynamic online nature of our system, i.e., the current scheduling decision and casualty will affect casualty performance of the future. Specifically, whether or not drone d is selected for transmission at time δ_k (current scheduling decision) will determine whether enemy $e_d(\delta_k)$ will be eliminated at time δ_k^+, which in turn will affect the probability

[1]Hector is the great ancient Trojan warrior.

for drone d to be destroyed at time $t > \delta_k^+$ (future casualty). This makes it impossible to obtain a closed form expression for the casualty of drones (objective function).

Second, one might consider whether it is possible to calculate the casualty numerically by enumerating all possible problem inputs (i.e., $A_{d(e)}(\delta_k)$ for all drones d's and enemies e's at all time δ_k's). This numerical approach is practically infeasible because the state space for the inputs is too large.

To address the above two problems, we propose to employ the model-free RL approach to solve our problem. Model-free RL is a machine learning technique that learns an unknown optimal model without prior problem modeling through function approximations. It does not require an explicit formulation for the casualty (objective function) and is suitable for solving problems with a very large input space. An RL algorithm (a neural network) trained properly can perform well even for input sets that are never been learned in the training phase. In the rest of this section, we show how to design an RL-based scheduler to minimize the casualty of drones.

3.2 Key Idea

At each time δ_k, given $A_{d(e)}(\delta_k)$ for all drones d's and enemies e's, the objective of Hector is to select one drone for image transmission such that the casualty of drones can be minimized by T that is the finishing time of the mission. Suppose an unknown optimal model exists for our problem, denoted by Π, that minimizes the casualty of drones. Then, the basic idea behind Hector is to find an approximation μ of this optimal model Π without prior problem modeling. After the training phase, μ will have the ability to provide predictions of decisions of Π even when the input is outside the training dataset [10].

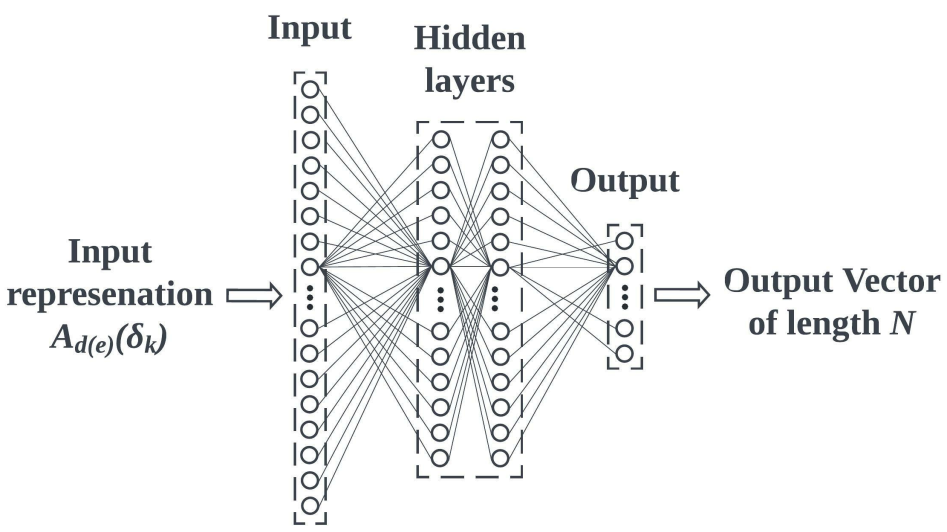

Input
Hidden layers
Output
Input represenation $A_{d(e)}(\delta_k)$
Output Vector of length N

Figure 3.1: Structure of neural network μ.

In particular, μ is a neural network with numerous trainable parameters within its structure that determines the mapping from the input to the output. Fig. 3.1 shows the structure of μ. The input to μ is $A_{d(e)}(\delta_k)$ for all drones d's and enemies e's. The output from μ is a vector of length N, with entry $i \in \{1, 2, \cdots, N\}$ representing the probability to select the i-th drone for transmission at time δ_k. We build fully-connected layer structures for μ (i.e., a neural network with two hidden layers, where each layer has 256 neurons) so that a strong universal function approximating capability can be offered.

Training of μ is the core component of Hector, which has multiple steps. The goal of training is to optimize parameters in μ. Our training method has an actor-critic structure [20, 22]. In the next section we present the details of our training method.

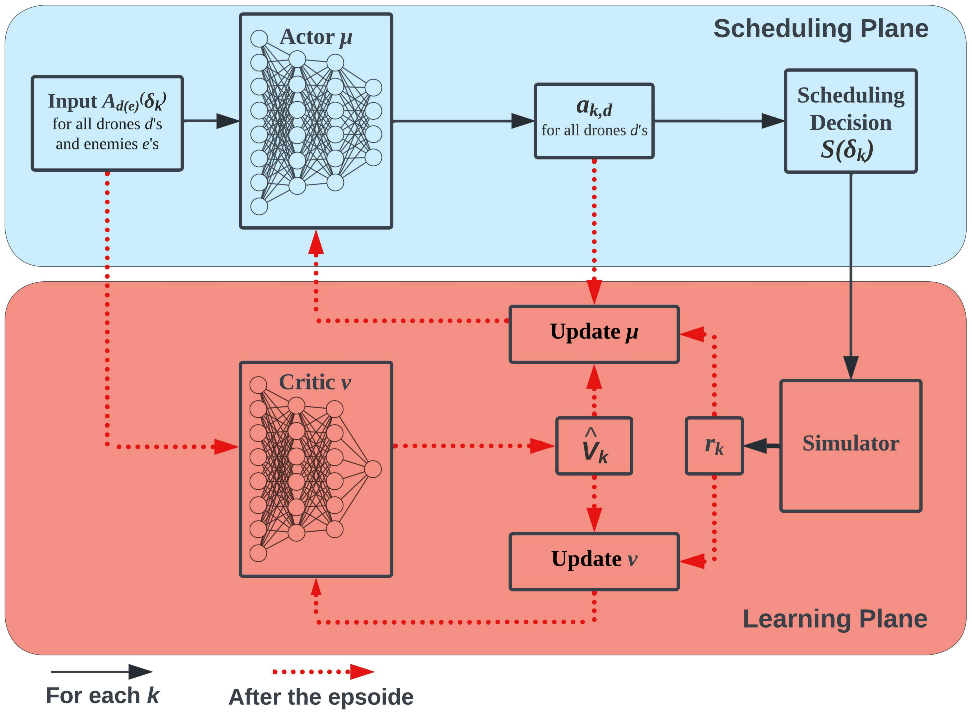


Figure 3.2: Structure of the training process for neural network μ.

3.3 Training Details

The actor-critic method is a policy gradient algorithm that consists of one actor and one critic. The BS collects input information and makes decisions through the actor, while the critic evaluates decisions made by the scheduler and optimize the actor accordingly. In this paper, we use the approach Proximal Policy Optimization (PPO) [24] to train the actor and critic, which has advantages of robustness and insensitivity to hyperparameters.

Fig. 3.2 shows the structure of this actor-critic training process, which consists of a scheduling plane and a learning plane. The neural network μ is considered as the actor, while another neural network ν is considered as the critic and shares the same input and structure of hidden layers as μ. The learning plane is responsible to optimize parameters in μ when one episode is completed. The scheduling plane is responsible to use μ to schedule one drone for image transmission at each time δ_k in one episode. During training, the two planes work together through a series of steps to optimize μ (in terms of approximating the optimal algorithm π).

Scheduling Plane: Denote K as the number of scheduling decisions to be made within one episode (e.g., $K = 2^{12}$ in our training). At time δ_k ($k = 1, 2, \cdots, K$) in an episode, the input to the actor is $\{A_{d(e)}(\delta_k), \text{for all drones } d's \text{ and for all enemies } e's\}$. The output from the actor is $\{a_{k,d}$ for all drones d's$\}$, where $a_{k,d}$ is the probability for the scheduler to select drone d for transmission at time δ_k. Then, the scheduler choose the drone for transmission based on the probability distribution of $a_{k,d}$'s.

Learning Plane: At time δ_k, based on the scheduling decision $\boldsymbol{S}(\delta_k)$ made by the actor in the scheduling plane, the simulator calculates a reward r_k (to be defined in (3.1)) which measures the quality of $\boldsymbol{S}(\delta_k)$. Then when the entire episode is completed, the critic takes $A_{d(e)}(\delta_k)$ as inputs and outputs $\hat{V}_k$ for all $k = 1, 2, \cdots, K$. The actor is updated by Adam optimizer [19] based on r_k's and $\hat{V}_k$'s along the direction defined by a loss function $L_\mu^{(j)}$ (see (3.2)). Likewise, the critic is updated by Adam optimizer based on r_k's and $\hat{V}_k$'s along the direction defined by a loss function $L_\nu^{(j)}$ (see (3.6)).

During the offline training process, each action a_k's results in a reward r_k, which we define as follows:

$$r_k = -\left(\sum_d p_d(\delta_{k+1}) - \sum_d p_d(\delta_k)\right). \tag{3.1}$$

where $p_d(\delta_k)$ is the risk probability for drone d to be shot down at time δ_k and is defined in (1.4). By definition (3.1), r_k is the difference between the expected casualty at time δ_{k+1} and that at time δ_k, where we assume that none of drones will be shot down during the time interval $[\delta_k, \delta_{k+1}]$. Note that although $p_d(\delta_{k+1})$ is considered as future information, it is not a problem as the training process is done offline and information for all enemies e's at any time $t \in [1, T]$ can be generated by the simulator. Also, note that the greater the r_k, the smaller the $(\sum_d p_d(\delta_{k+1}) - \sum_d p_d(\delta_k))$, which means smaller total risk probability at time δ_{k+1}. Intuitively, the reward r_k tends to incentivize the actor μ if it can produce an output (a_k) that reduces total risk probabilities at the next scheduling time instance. Note that once the offline training process is completed, our algorithm Hector (the upper scheduling plane in Fig. 3.2) will operate as an online algorithm.

The critic is a neural network that evaluates the performance of the actor. It estimates how the action made based on $A_{d(e)}(\delta_k)$'s at time δ_k affects the casualty of drones in the future, i.e., future time interval of $[\delta_k, \delta_K]$. The output of the critic, $\hat{V}_k$, attempts to estimate $\sum_{i=0}^{K-k} \gamma^j \cdot r_{k+i}$ where γ $(0 < \gamma < 1)$ is a discount factor. $\sum_{i=0}^{K-k} \gamma^i \cdot r_{k+i}$ is the cumulative reward from time δ_k to time δ_K. For easy of reference, we define $V_k = \sum_{i=0}^{K-k} \gamma^j \cdot r_{k+i}$. Each time when the entire episode is completed, we calculate $\hat{V}_k$ and V_k for all $k = 1, 2, \cdots, K$.

Each time when an episode is completed, we use Adam optimizer to update actor μ for J times (e.g., 16 in our training). For the j-th update $(j = 1, 2, \cdots, J)$ of μ, we employ a loss function $L_\mu^{(j)}$ which is defined as follows:

$$L_\mu^{(j)} = -\mathbb{E}\left[\min\left\{q_k^{(j)} \cdot G_k,\ \text{clip}\left(q_k^{(j)}, 1-\epsilon, 1+\epsilon\right) \cdot G_k\right\}\right], \tag{3.2}$$

where $q_k^{(j)}$ is a ratio of probabilities and is defined as follows:

$$q_k^{(j)} = \frac{a_{k,d}^{(j-1)}}{a_{k,d}}, \text{ where } d = \{i :\ S_i(\delta_k) = 1\}. \tag{3.3}$$

In (3.3), $a_{k,d}$ is the output of actor μ corresponding to drone d that is selected by μ for transmission at time δ_k, and $a_{k,d}^{(j-1)}$ is the output of actor μ corresponding to $a_{k,d}$ after the $(j-1)$-th update of μ by Adam optimizer.

In (3.2), G_k is defined as:

$$G_k = \sum_{i=0}^{K-k-1} (\lambda \cdot \gamma)^i \cdot (r_{k+i} + \gamma \cdot \hat{V}_{k+i+1} - \hat{V}_{k+i}), \tag{3.4}$$

where λ $(0 \leq \lambda \leq 1)$ is a hyperparameter. G_k is used to measure the difference between estimated cumulative rewards $\hat{V}_k$ and observed cumulative rewards V_k and is calculated according to Generalized Advantage Estimation (GAE) [25].

The function clip is defined as follows:

$$\text{clip}\left(q_k^{(j)}, 1-\epsilon, 1+\epsilon\right) = \begin{cases} 1-\epsilon, & \text{if } q_k^{(j)} < 1-\epsilon; \\ 1+\epsilon, & \text{if } q_k^{(j)} > 1+\epsilon; \\ q_k^{(j)}, & \text{otherwise,} \end{cases} \tag{3.5}$$

where $\epsilon > 0$ is a hyperparameter.

Each time when an episode is completed, we use Adam optimizer to update μ for J times. For the j-th update $(j = 1, 2, \cdots, J)$, we first calculate $L_\mu^{(j)}$ and then update μ along the direction defined by $L_\mu^{(j)}$ (similar to gradient descent approach). Such an update has the objective of maximizing G_k (hence V_k as G_k measures the difference between V_k and $\hat{V}_k$). The reason why we consider the loss function $L_\mu^{(j)}$ instead of G_K directly is because Schulman *et al.* in [24] proved that optimizing $L_\mu^{(j)}$ can make the training process faster and more stable than optimizing G_k.

Similarly, each time when the entire episode is completed, we use Adam optimizer to update critic ν for J times. For the j-th update of ν $(j = 1, 2, \cdots, J)$, we employ a loss function

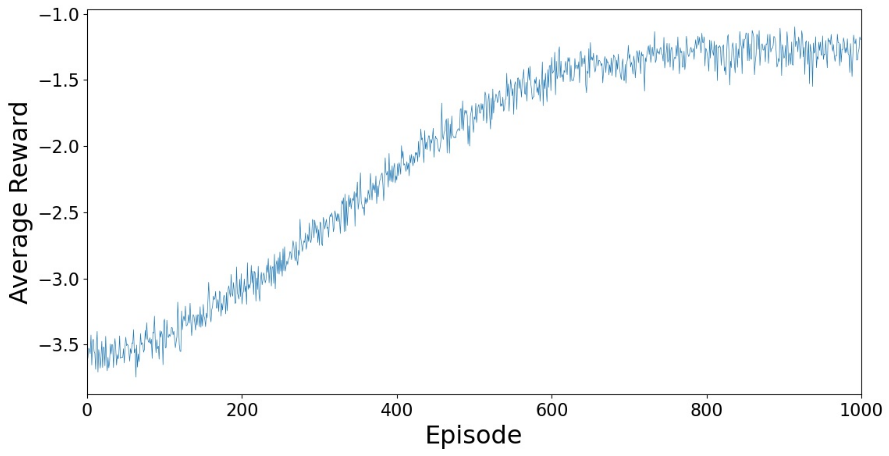


Figure 3.3: The change of average rewards $\frac{\sum_{k=1}^{K} r_k}{K}$ and casualties through episodes.

$L_\nu^{(j)}$ which is defined as follows:

$$L_\nu^{(j)} \;=\; \mathbb{E}\left[\left(\hat{V}_k^{(j-1)} - V_k\right)^2\right], \tag{3.6}$$

where V_k is the cumulative reward and $\hat{V}_k^{(j-1)}$ is the output of critic ν corresponding to $\hat{V}_k$ after the $(j-1)$-th update of ν by Adam optimizer. For the j-th update $(j = 1, 2, \cdots, J)$, we first calculate $L_\nu^{(j)}$ and then update ν along the direction defined by $L_\nu^{(j)}$. Such an update allows $\hat{V}_k^{(j)}$, the output of the updated ν, to better estimate V_k.

After training, at each time δ_k, Hector uses ages $A_{d(e)}(\delta_k)$ from all drones d's and enemies e's as inputs to μ and μ gives output $a_{k,d}$ for all drones d's. Then, Hector will select the drone corresponding to the largest $a_{k,d}$ for transmission at time δ_k.

3.4 Implementation and Evaluation

We implement Hector using TensorFlow [1] and Keras [7]. Fig. 3.3 shows the learning curves of our proposed training (Fig. 3.2), where each episode contains $K = 2^{12}$ steps. From the figure we observe that the learning process converges smoothly.

Chapter 4

Performance Evaluation

4.1 Evaluation Plan

In this section, we evaluate the performance of Hector. First, we use a case study to demonstrate how the casualty achieved by Hector evolves with time, and compare it against the performance of two benchmark algorithms: (i) our proposed greedy scheduler (requiring global knowledge), and (ii) a simple round-robin scheduler (not requiring global knowledge). Then, we study the performance behavior of Hector under varying system parameters.

Table 4.1: Simulation parameters for a drone swarm.

Parameters	Value
Number of drones in the array N	100
Speed of a drone	10 m/s
Total length of flight path	6 km
Radius of each drone's visual area R_d	200 m
Size of an image to be sent to the BS	625 KB
Data rate of uplink wireless channel	50 Mbps
Radius of each enemy's firing area R_e	200 m
Slope c of the function $f(\cdot)$ in (1.5)	0.001
Target generation probability π	0.1

4.2 Simulation Setup and Parameter Setting

Table 4.1 shows the general parameters in our experiment, which follows the scenario in Fig. 1.1. We set $N = 100$, i.e., 100 drones in an array. We set the space between two adjacent drones to 400 m. We assume the speed of each drone is 10 m/s and the total length of a drone's flight path is 6 km. So it takes 600 seconds to complete the flight mission and the coverage area by the 100 drones is 40km $\times$ 6km. For this coverage area, we further divide it into 100×600 grids, with each grid corresponding to a small area of 400m $\times$ 10m. Within each grid, we generate a target following a Bernoulli distribution with a success probability of π (i.e., with probability of π a target exists in this grid). π is also called *target generation probability* in our simulation study. If there is a target in a grid, we assume its location following a uniform distribution inside the grid. We set $\pi = 0.1$ in the experiment unless otherwise specified. Further, we assume the probability that the target is an enemy is 0.25 and the probability that the target is a civilian is 0.75, respectively.

As shown in Table 1.1, we set the radius of a drone's visual area to be $R_d = 200$ m. We assume images to be sent to the BS share the same size of 625K bytes. We set the uplink data rate to 50 Mbps. Hence the time for transmitting one image from a drone to the BS is 100 ms (i.e., 625KB/50Mbps).

Since we will employ discrete event simulation in our experimental study, we need to discretize time so that we can advance the simulation following a sequence of time intervals. We set the smallest time interval to be 100 ms, same as the time needed to transmit an image from a drone to the BS.

On the enemy side, we assume that the function $f(\cdot)$ (which is used to calculate the risk probability $\alpha_{e(d)}(t)$) follows the linear function as (1.5), with $c = 0.001$ in the experiment unless otherwise specified. We set the radius of an enemy's firing area to be $R_e = 200$

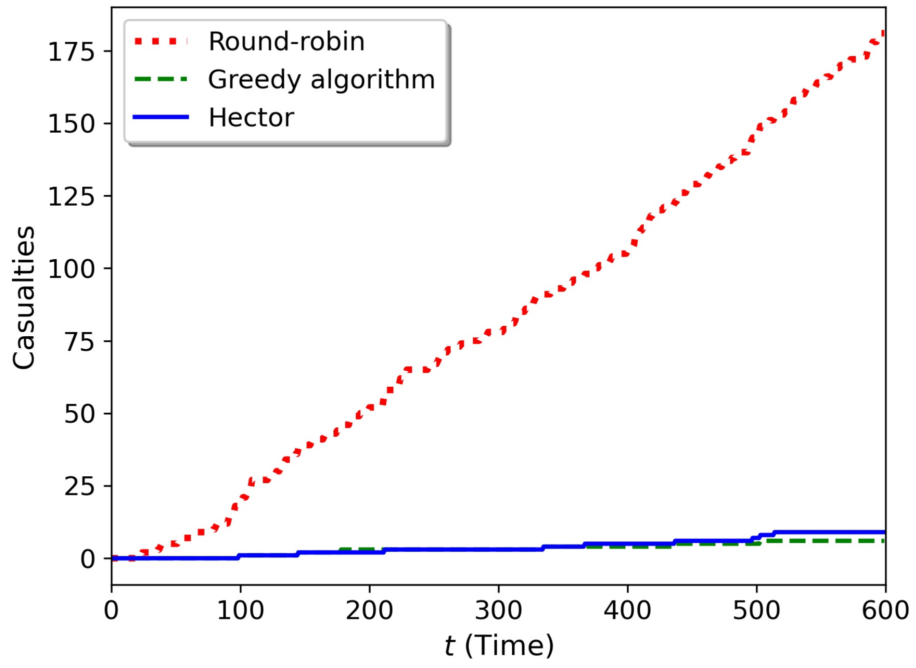


Figure 4.1: Evolution of casualty over time t under different scheduling algorithms.

m unless otherwise specified. At time instances of $t = 1, 2, \cdots, 600$ s, each enemy e will decide whether or not to shoot down a drone in its firing area with a risk probability $\alpha_{e(d)}(t)$ (defined in (1.3)).

4.3 A Case Study

We use a case study to illustrate the performance (casualties) of Hector over time and compare it to the greedy algorithm and round-robin algorithm. Fig. 4.1 shows this result.

From the figure, we observe that by the end of the mission ($T = 600$ s), 181 drones are destroyed under the round-robin scheduler, 6 drones are destroyed under the greedy scheduler, and 9 drones are destroyed under Hector. In this case study, we find that both the greedy scheduler and Hector significantly outperform the round-robin scheduler in terms of fewer casualties, indicating the necessity of a well-designed scheduler to address our problem. Further, the performance of Hector is close to that of the greedy scheduler, even though Hector only employs local information while the greedy scheduler requires global knowledge.

4.4 Varying System Parameters

4.4.1 Varying Load (Target Generation Probabilities)

Now we evaluate the performance of Hector by varying the number of targets (load) along the flight path. This is done by varying the target generation probability π from 0 to 0.16, with an increment of 0.01. For each π value, we simulate 100 instances and take the average of casualty results. The results of this study are shown in Fig. 4.2.

From Fig. 4.2 we observe that when $\pi \in [0.03, 0.16]$, the casualties achieved by Hector are almost identical to those achieved by the greedy scheduler. The casualties achieved by Hector and the greedy scheduler are lower than those achieved by the round-robin scheduler, especially when $\pi \in [0.04, 0.12]$. An interesting observation is that when $\pi \leq 0.1$, the casualties of Hector and the greedy algorithm remain close to 0. But when $\pi > 0.1$, the casualties of Hector and the greedy algorithm start to ramp up. This is because $\pi = 0.1$ implies that the expected number of targets in a grid is 0.1. Considering that 100 drones form an array. With a flying speed of 10 m/s, a drone will cover 1 grid per second. So the number of targets detected by the drone swarm will be $100 \cdot 0.1 = 10$ per second. Since a drone takes a picture every 100 ms, the drone swarm is expected to detect one target every 100 ms. Since the time for transmitting one image is also 100 ms, it is expected that the information of every target that appears in the territory will be transmitted to the BS and

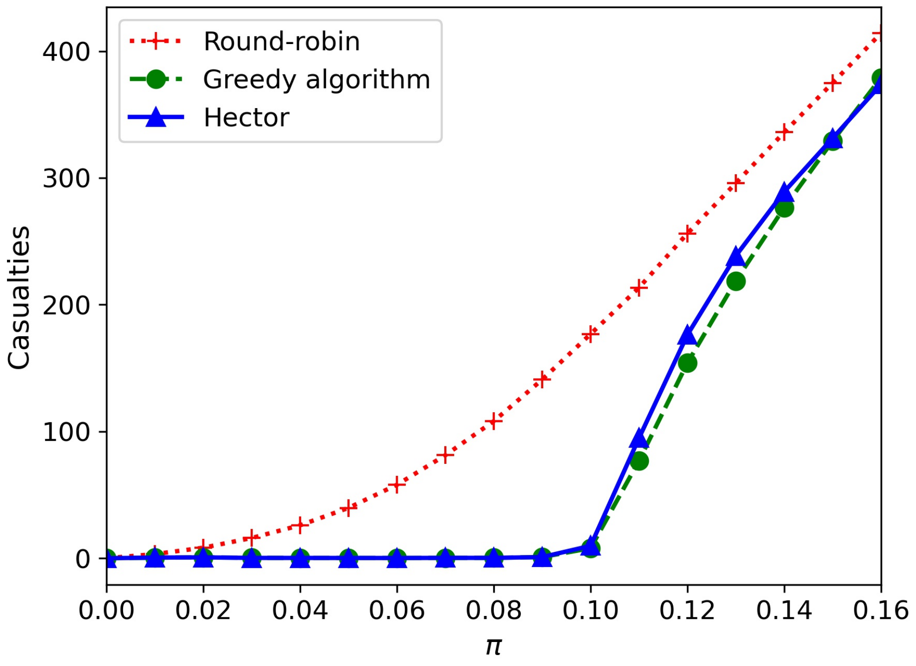


Figure 4.2: Casualty results by different scheduling algorithms for different values of π.

hence every enemy will be eliminated by the drone swarm, leading to a casualty close to 0. When $\pi > 0.01$, the drone swarm is expected to detect more than one target every 100 ms. That is, there is more images containing enemies than the drones can upload and take lethal action in time. Consequently, there is an increase of risk probabilities for the drones, leading to more casualties.

Not shown in Fig. 4.2 are the casualty performance for the three schedulers when π is larger than 0.16. We find that the casualties of different schedulers all converge eventually when π is larger than 0.18. This is intuitive because under such heavy "load" (target generation probability), there are simply too many enemies exist in the territory for the drones to handle and a better performance is not achievable through the design of a scheduler.

From Fig. 4.2 we also observe when $\pi < 0.03$, the casualties achieved by different schedulers

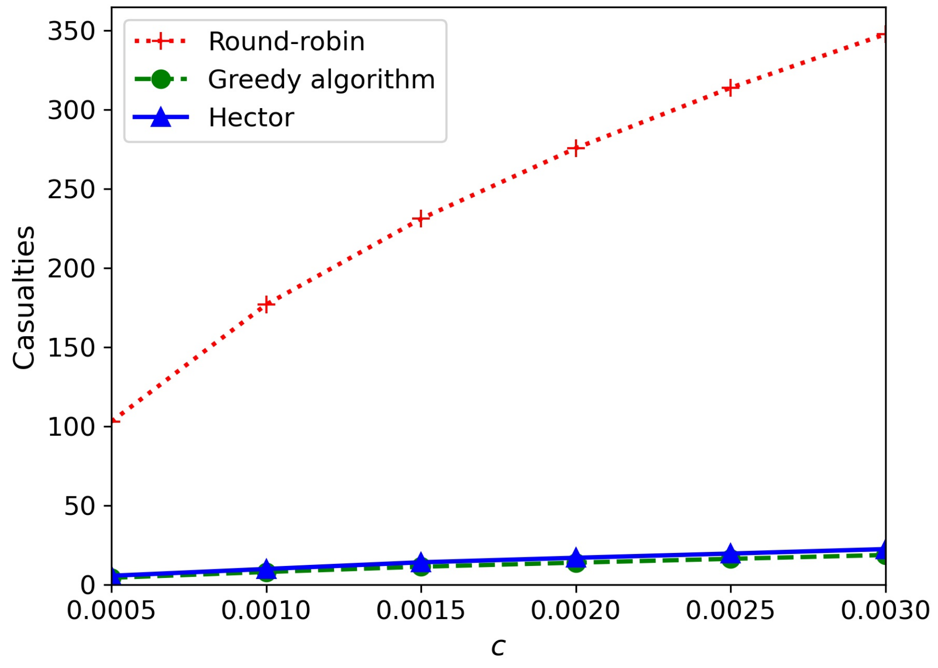


Figure 4.3: Casualty results by different scheduling algorithms for different values of c.

are similar and all are close to 0, which is intuitive because under such low target generation probabilities, there are very few enemies exist along the flight path.

4.4.2 Varying Risk Probability

We evaluate the performance of Hector by varying the risk probability $\alpha_{e(d)}(t)$ in (1.3). This is done by varying c, the slope of the function $f(\cdot)$ in (1.5). For each value of c, we simulate 100 topology instances and take the average of casualty results. The results of this study are shown in Fig. 4.3.

From Fig. 4.3 we observe that when c varies from 0.0005 to 0.003, the casualties by the round-robin, Hector, and the greedy algorithm all increase, as expected; the performance of Hector is similar to the greedy algorithm; both Hector and the greedy algorithm perform

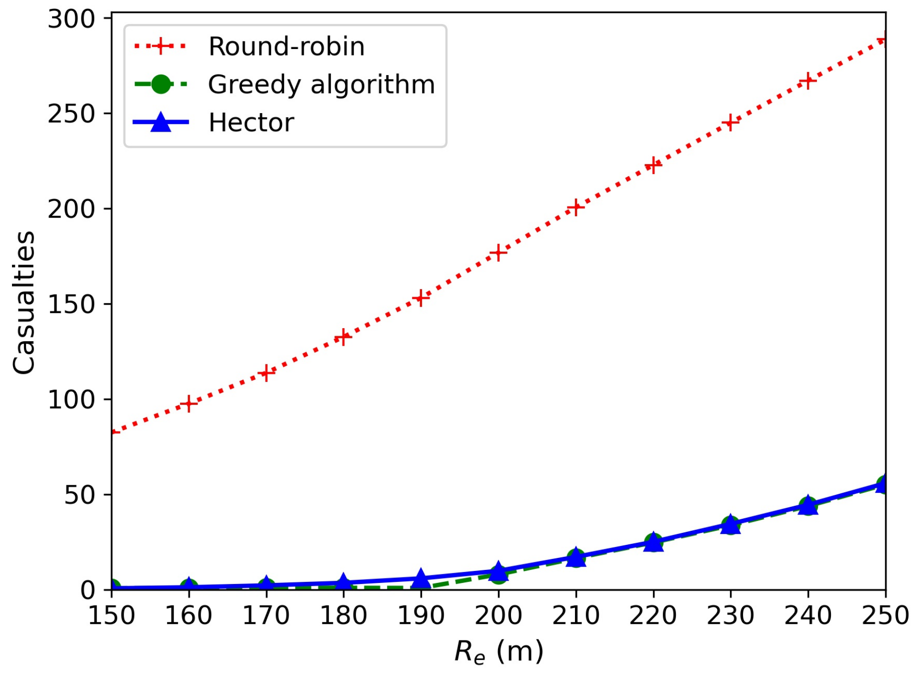


Figure 4.4: Casualty results by different scheduling algorithms for different values of R_e.

much better than the round-robin algorithm. Specifically, when $c = 0.0005$, the number of casualties by round-robin, Hector, and the greedy algorithms are 102.86, 5.51, and 4.23, respectively. When c is increased to 0.0030, the number of casualties by round-robin, Hector, and the greedy algorithms become 347.58, 22.36, and 18.63, respectively.

4.4.3 Varying Enemy's Firing Range

Finally we evaluate the performance of Hector by varying R_e—the radius of each enemy e's firing area. For each value of R_e, we simulate 100 topology instances and take the average of casualty results. The results of this study are shown in Fig. 4.4.

From Fig. 4.4 we observe that when R_e varies from 150 to 250, the casualties by the round-robin, Hector, and the greedy algorithm all increase, as expected; the performance of Hector is similar to the greedy algorithm; both Hector and the greedy algorithm perform much better than the round-robin algorithm. Specifically, when $R_e = 150$ m, the average the number of casualties by round-robin, Hector, and the greedy algorithms are 82.39, 0.61, and 0.71, respectively. When R_e is increased to 250 m, the number of casualties by round-robin, Hector, and the greedy algorithms become 288.66, 55.81, and 55.15, respectively.

Chapter 5

Conclusions

In this thesis, we investigate a problem of scheduling transmissions from a drone swarm with an objective of minimizing its casualties. For this problem, we first design a greedy scheduling algorithm. At each transmission opportunity, this greedy algorithm transmits the drone whose transmission leads to the largest reduction in its probability of being destroyed by enemies. From extensive simulations we observe that this greedy algorithm significantly reduces casualties than a round-robin baseline. However, this greedy algorithm has a limitation that it requires the drone swarm to have full knowledge of enemies' information to make scheduling decisions, which is not possible in reality. To address this limitation, then we develop an efficient online scheduling algorithm Hector. Hector uses the age locally available at the drones as inputs to a neural network and exploits reinforcement learning techniques to make scheduling decisions. Extensive simulations indicate that the performance of Hector is always close to that of the greedy algorithm, despite that the greedy algorithm requires the knowledge of certain information that is not available in practice while Hector does not have such a requirement.

Bibliography

[1] Martín Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] M.A Abd-Elmagid and H.S Dhillon. Average peak age-of-information minimization in UAV-assisted IoT networks. *IEEE Transactions on Vehicular Technology*, 68(2):2003–2008, February 2019.

[3] Ghafour Ahani, Di Yuan, and Yixin Zhao. Age-optimal UAV scheduling for data collection with battery recharging. *IEEE Communications Letters*, 25(4):1254–1258, April 2021.

[4] Asisguard. Songar drone systems. [Online]. Available: https://asisguard.com.tr/en/product/songar-armed-with-gimbal/. [Accessed: June 5, 2022].

[5] C.M.W Basnayaka, D.N.K Jayakody, and Zheng Chang. Age of information based URLLC-enabled UAV wireless communications system. *IEEE Internet of Things Journal*, 2022, to appear.

[6] M.H Cheung. Age of information aware UAV network selection. In *Proc. of WiOpt*, 8 pages, virtual conference, June 15–19, 2020.

[7] Francois Chollet et al. Keras. [Online]. Available: https://github.com/fchollet/keras. [Accessed: June 5, 2022].

[8] Gioele Ciaparrone, F.L Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88, March 2020.

[9] General Atomics. Mulit-mission remotely piloted aircraft. [Online]. Available: https://www.ga-asi.com/products-services/. [Accessed: June 5, 2022].

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. ISBN 9780262035613.

[11] Rui Han, Jiaxing Wang, Lin Bai, Jianwei Liu, and Jinho Choi. Age of information and performance analysis for UAV-aided IoT systems. *IEEE Internet of Things Journal*, 8 (19):14447–14457, October 2021.

[12] Huimin Hu, Ke Xiong, Gang Qu, Qiang Ni, Pingyi Fan, and K.B Letaief. AoI-minimal trajectory planning and data collection in UAV-assisted wireless powered IoT networks. *IEEE Internet of Things Journal*, 8(2):1211–1223, January 2021.

[13] Jingzhi Hu, Hongliang Zhang, Kaigui Bian, Lingyang Song, and Zhu Han. Distributed trajectory design for cooperative internet of UAVs using deep reinforcement learning. In *Proc. of IEEE GLOBECOM*, 6 pages, Big Island, Hawaii, USA, December 9–13, 2019.

[14] Licheng Jiao, Dan Wang, Yidong Bai, Puhua Chen, and Fang Liu. Deep learning in visual tracking: A review. *IEEE Transactions on Neural Networks and Learning Systems*, early access, 2021.

[15] Heng Jin, Qingyu Liu, and Y.T Hou. Tracking, control, and optimization of information latency for military drone swarm. [Online]. Available: https://www.youtube.com/watch?v=rcUts3m-eao. [Accessed: May 12, 2022].

[16] M.A Kassab, Ali Maher, Fathy Elkazzaz, and Zhang Baochang. UAV target tracking by detection via deep neural networks. In *Proc. of IEEE ICME*, pp. 139–144, Shanghai, China, July 8–12, 2019.

[17] Sanjit Kaul, Marco Gruteser, Vinuth Rai, and John Kenney. Minimizing age of information in vehicular networks. In *Proc. of IEEE SECON*, pp. 350–358, Salt Lake City, UT, USA, June 27–30, 2011.

[18] Sanjit Kaul, Roy Yates, and Marco Gruteser. Real-time status: How often should one update? In *Proc. of IEEE INFOCOM*, pp. 2731–2735, Orlando, FL, USA, March 25–30, 2012.

[19] D.P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. In *Proc. of NIPS*, pp. 1008–1014, Denver, CO, USA, November 29–December 4, 1999.

[21] Lockheed Martin. Autonomous and unmanned systems. [Online]. Available: https://www.lockheedmartin.com/en-us/capabilities/autonomous-unmanned-systems.html/. [Accessed: June 5, 2022].

[22] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proc. of ICML*, pp. 1928–1937, New York, NY, USA, Jun 20–22, 2016.

[23] Parrot. Use case: Military & defense ANAFI USA. [Online]. Available: https://www.parrot.com/uk/use-cases/military-and-defense/. [Accessed: June 5, 2022].

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

[25] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel.

High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2018.

[26] P.J Springer. *Military Robots and Drones : A Reference Handbook*. ABC-CLIO, 2013. ISBN 9781598847321.

[27] Yin Sun. A collection of recent papers on the age of information. [Online]. Available: http://webhome.auburn.edu/~yzs0078/. [Accessed: April 7, 2022].

[28] U.S. Department of Defense. Unmanned aircraft systems (UAS) DoD purpose and operational use. [Online]. Available: https://dod.defense.gov/UAS/. [Accessed: June 5, 2022].

[29] Shuhang Zhang, Hongliang Zhang, Zhu Han, H.V Poor, and Lingyang Song. Age of information in a cellular internet of UAVs: Sensing and communication trade-off design. *IEEE Transactions on Wireless Communications*, 19(10):6578–6592, October 2020.

[30] Xi Zhang, Jingqing Wang, and H.V Poor. Statistical delay and error-rate bounded QoS provisioning for 6G mURLLC over AoI-driven and UAV-enabled wireless networks. In *Proc. of IEEE INFOCOM*, 10 pages, virtual conference, May 10–13, 2021.