electronics

MDPI

Article

Reservation-Based 3D Intersection Traffic Control System for Autonomous Unmanned Aerial Vehicles

Areeya Rubenecia [1], Myungwhan Choi [1] and Hyo-Hyun Choi [2,*]

[1] Department of Computer Science and Engineering, Sogang University, Seoul 04107, Korea; rubenecia@sogang.ac.kr (A.R.); mchoi@sogang.ac.kr (M.C.)
[2] Department of Computer Science and Engineering, Inha Technical College, Incheon 22212, Korea
* Correspondence: hchoi@inhatc.ac.kr

Abstract: We present a three-dimensional (3D) intersection traffic management platform for small autonomous Unmanned Aerial Vehicles (UAVs), particularly quadcopters, in urban airspace. Assuming many autonomous UAVs are approaching a shared airspace, where UAVs have varying sources and destinations, we propose a system model for a 3D intersection that aims to provide safe and systematic management of UAVs. We also devised a scheduling scheme to ensure that the intersection is efficiently utilized and that there are no collisions among the UAVs in the intersection. The scheduling scheme applies the reservation-based approach, which is sensitive to the sequence of the UAVs in scheduling, thus genetic algorithm is used to determine the best sequence of the UAVs. Simulations were performed to evaluate the efficiency of the system. We also show through the simulations that our scheduling scheme reduces the UAVs' average time in the system by 27 percent compared with when the UAVs are scheduled in a first-come, first-served manner for the highly crowded intersection.

Keywords: unmanned aerial vehicles (UAVs); 3D intersection; intersection traffic control; reservation-based scheduling; genetic algorithm

Citation: Rubenecia, A.; Choi, M.; Choi, H.-H. Reservation-Based 3D Intersection Traffic Control System for Autonomous Unmanned Aerial Vehicles. *Electronics* **2022**, *11*, 309. https://doi.org/10.3390/electronics11030309

Academic Editor: Zhiwei Gao

Received: 9 November 2021
Accepted: 16 January 2022
Published: 19 January 2022

1. Introduction

UAVs have been gaining interest due to their applications in various areas such as weather monitoring, law enforcement, agriculture, search and rescue, and communication networks [1,2]. The increasing use of UAVs calls for a system that will ensure safety and efficiency in the airspace [3]. Some laws and regulations for the Unmanned Aircraft System Traffic Management (UTM) framework have been established to improve the safety, security, and efficiency of small UAVs [4].

In the ground transportation system, rules and conventions such as roads, stop signs, traffic signals, intersections, etc. are implemented to provide a proper level of organization to various vehicles. Inspired by this, we proposed an intersection structure in the sky where multiple UAVs can travel across an intersection safely. Our work can be used on applications that require the use of quadcopters such as delivery, surveillance, search and rescue, etc.

The scope of our system includes the management of the UAVs' flights from when they approach until they exit the intersection. The UAVs are instructed to follow given rules to be able to travel the intersection safely. The paths of the UAVs in the intersection are centrally managed and cooperatively planned. An intersection manager plans the paths and entrance times of the UAVs in the intersection. Genetic algorithm and modified A* search are used to plan the paths of the UAVs that will minimize their travel times. A reservation-based scheduling approach is used to find the collision-free path in the intersection.

The study in [5] introduced the reservation-based scheduling approach to ensure safety in the intersection, which we applied in our scheduling scheme. In [5], the intersection is

divided into multiple sections called cells. When a vehicle travels along the intersection, it will occupy certain cells for a certain time. To avoid collision among the vehicles in the intersection, a cell can be reserved to only one vehicle at a time. There is an intersection manager that has complete information on the reservations of every cell. Before a vehicle enters the intersection, it will send its request to occupy some cells in the intersection to the intersection manager. The intersection manager will check the cells and times they are requested to be occupied and will make the reservation. In our proposed scheme, the intersection space is divided into 3D sections called cubes, and the UAVs reserve the cubes.

Our contribution in this work is that we devised a 3D intersection traffic management system to regulate multiple small autonomous quadcopter-type UAVs in shared airspace. To the best of our knowledge, there is no other proposed 3D intersection traffic management systems for autonomous UAVs with the same system model as ours. In particular, we designed a system model that describes the intersection area. We devised a scheme to guarantee that there will be no collisions among UAVs while they approach the intersection. We defined the allowed movements of the UAVs in the 3D intersection. To guarantee collision avoidance in the intersection, we implement the reservation-based scheduling introduced in [5]. We used Genetic algorithm to optimize the scheduling sequence. Since there are multiple possible paths in the intersection across different layers, path-finding is used to find the fastest path in the intersection. In our path-finding implementation, already checked nodes sometimes need to be checked again later. So, we developed a new path-finding algorithm based on the A* search and applied it to our scheduling algorithm.

This paper is organized as follows: Section 2 introduces the related works on path planning of multiple UAVs; Section 3 discusses the system model, which includes the intersection structure and the overview of how the system works; Section 4 explains how the UAVs should move before they enter the intersection; Section 5 describes the allowed trajectories of the UAVs in the intersection; Section 6 discusses the scheduling scheme; Section 7 includes the analysis of the scheduling scheme; Section 8 discusses the simulation results; Finally, Section 9 concludes the paper.

2. Related Work

The problem of UAVs' integration into the urban airspace was also addressed in papers in [6–9]. A framework consisting of a decentralized approach is implemented in [6,7]. In [8,9], a structure for the low-altitude airspace com of airways and nodes was proposed, where the UAVs travel along the airways and the airways are connected by the nodes. The UAVs decide which airways to take according to their objectives. The main difference between their approach with ours is that our approach is focused on the intersection level. In their proposal, they assume only one swarm of UAVs, with the same source and destination, can occupy a node at the same time. In our proposal, there can be multiple UAVs, with different sources and destinations, traveling across the intersection at a time.

Consequently, there have been many studies on the path-planning of UAVs. Among these studies, the common considerations are collision avoidance, kinematic constraints of the UAVs, and optimization of various criteria such as minimizing the path length and energy or fuel consumption. Collision avoidance is crucial for the safer integration of UAVs in the airspace. A comprehensive survey on recent collision avoidance approaches is in [10]. In this study, collision avoidance techniques are categorized into deliberative and reactive planning. In deliberative planning, a collision-free path is searched for in a known updated map of the environment and the found path is executed while in reactive planning, the UAVs gather information about the surroundings in real-time using sensors, and the UAVs react based on the obtained sensor information.

When planning a path in an unknown environment, sampling-based algorithms are used to generate a path from the starting position of the UAV to its destination. In a sampling-based approach, points in the environment are sampled and the points are added to the map if there is a collision-free path between the points [11]. Some of the

sampling-based approaches are rapidly exploring random trees [12,13], probabilistic road map [14,15], and Voronoi graphs [16]. To consider the feasibility of the paths based on the kinematic constraints of the UAVs, the paths are generated using Dubins curve in [13,17,18]. Dubins curve provides the fastest and shortest path for non-holonomic vehicles. In [19,20], Pythagorean hodograph curve is used while B-spline curve is used in [21–24], and Bezier curve is used in [25] to consider the minimum curvature, minimum torsion, and maximum climb angle of the UAV.

A path-searching algorithm is used to find the fastest path among the set of possible points on the map. In [26], they compared different path searching algorithms such as Dijkstra's algorithm, Bellman Ford's algorithm, Floyd-Warshall's algorithm, and A* algorithm, which is also used in [15,27,28]. In [14], probabilistic road map is used to plan the initial path of the UAV, then D* lite is used to remove the unnecessary points in the path. D* lite is a heuristic search algorithm introduced in [29]. In [16], the problem is target assignment and path planning of multiple UAVs that attack ground targets. K-shortest path is used to find the path among the candidate paths generated using Voronoi diagrams. In [30–32], the environment is known and the paths of the UAVs considering static obstacles are planned offline and then online re-planning is implemented to change their trajectories when dynamic obstacles are detected. In [33], an enhanced artificial potential field approach, which aims to overcome the limitation of the conventional artificial potential field, is introduced to select the optimal collision-free path.

In relation to reactive online path planning, the works in [34–40] use an online collision avoidance algorithm that relies on the sensors of the UAVs to detect the obstacles. In [34], when an obstacle is detected, an escape point algorithm is used to find a waypoint that will avoid the obstacle, and then the UAV will move towards the goal again after reaching the waypoint. To detect obstacles in [35], a total field sensing approach is used that uses magnetic sensors to detect other UAVs. In [36], a decentralized reactive algorithm was proposed. The concept of velocity obstacle is used in [37] to find a path away from the detected obstacle. In [38], a decentralized cooperative control scheme was devised. UAV heartbeat messages are used to enable cooperative communication and velocity obstacle is used to avoid obstacles. In [39], a geometrical intersection method is used to estimate a collision risk and new direction commands are generated for every UAV at risk for collision. An interval geometric formulation is used in [40] for collision avoidance with multiple dynamic obstacles.

Many path planning approaches also use optimization methods such as particle swarm optimization [19,41], ant colony optimization [42,43], genetic algorithm [16,18,25,43–45], evolutionary algorithms [22–24], and MILP [46–48] to find optimal paths. In [43], ant colony and genetic algorithm are used to find a path considering sensing, energy, time, and risk constraints. In [16], genetic algorithm is used to minimize the total time to finish the tasks. In [18], the objective is to generate the shortest Dubins path. Communication among multiple UAVs for search and rescue missions is one of the objectives in [45]. In [22–24], an evolutionary algorithm is used to find a path that considers the feasibility, length, and safety of the path. On the other hand, since there is a tradeoff between the optimality of the paths and the speed of the computation [49], the optimality of the trajectories is not considered in [36]. Instead of planning the paths of the UAVs in advance, they prioritize safety and low computational overhead, which can be applied in a large-scale harsh outdoor environment. In [46–48], they used MILP to find a collision-free path with minimal total time spent by the UAV.

A common approach in coordinating multiple UAVs is swarm formation [6,10]. In a UAV swarm, multiple UAVs work together to achieve a common mission. Swarm formation of UAVs is usually used in search and rescue missions, tracking, surveillance, and object detection [50]. Communication is crucial in UAV swarm control [51]. The work in [52] controls a swarm of UAVs using mean field game framework to reduce the necessity of communication between UAVs. In [53], a leader-follower based strategy was used to control

the swarm formation. A swarm of UAVs was also used in [54] to implement a search and rescue mission. Limited communication range is considered in their model.

3. System Model

Considering a situation where there are many flying UAVs with different destinations in shared airspace, an intersection system model is developed to regulate the flights of UAVs. We assume UAVs travel in urban airspace that is free of obstacles and other aircraft. We consider a 3D intersection with three layers. Connected to the intersection are entrance lanes and exit lanes from four different directions. The UAVs travel along the lanes and move towards the intersection. The lanes are connected to the middle layer of the intersection, thus UAVs can only enter and exit the intersection through the middle layer, but they can change layers while in the intersection. This model can be used for any number of lanes and can be extended to more layers.

The UAVs in consideration are small quadcopters, which have the ability to hover [55], to move upward, downward, leftward, and rightward in a curved motion smoothly. In this paper, a UAV is represented as a sphere. It can be of any diameter smaller than the width of the lane. Since it is impractical to assume that all UAVs move at a constant speed, we have a speed allowance and a UAV's speed can be any value within $[s_{min}, s_{max}]$. The UAVs must be able to follow the rules, and we assume they can communicate with the UAVs ahead and with the intersection manager without communication delay and there are no transmission errors. A UAV must communicate with the UAV ahead so that it can accordingly plan its speed that will avoid collision with the UAV ahead.

The paths of the UAVs in the intersection are scheduled by the intersection manager. While a UAV is approaching the intersection, it needs to prepare for its entrance to the intersection and it needs to control its speed so that it will enter the intersection as scheduled. To give time for a UAV to prepare for its entrance to the intersection, a section of the lane near the intersection, defined as approaching area, is partitioned into zones, namely, the reservation zone, queueing zone, and acceleration zone. A UAV must send its reservation request once it enters the reservation zone and it must receive its schedule before it enters the next zone, the queueing zone. In the queueing zone, it must slow down when it needs to delay its entrance to the intersection to be able to follow its schedule. It must stop before the entrance of the acceleration zone in case it needs to stop and wait before it can enter the intersection. It can enter the acceleration zone such that it will enter the intersection as scheduled. Figure 1 shows the 3D intersection model.

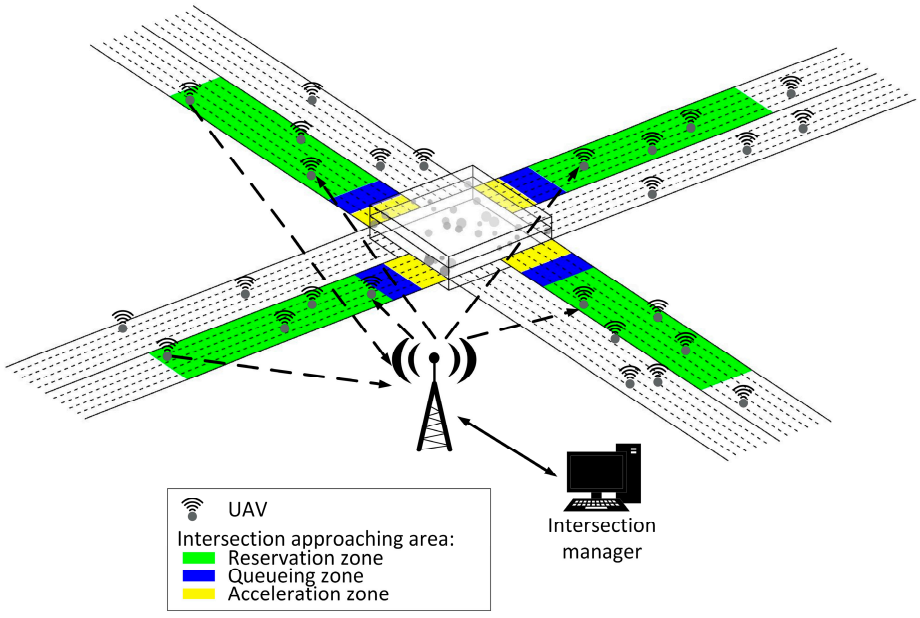


Figure 1. 3D intersection model.

The time is discretized using time interval Δt. At every time instant t_j where $t_j - t_{j-1} = \Delta t$, the UAVs on the lane decide the acceleration or deceleration rates that they will respectively use at t_{j+1} , such that they will not collide with a UAV ahead and they will enter the intersection as scheduled, if they are already scheduled. The requests of the UAVs are processed by the intersection manager at every time interval $k\Delta t$ where k is some positive integer and the scheduling time instant is synchronized with the discretized time. At scheduling time instant t_i, the requests processed by the intersection manager are the requests received at $(t_i - k\Delta t,\ t_i)$. After scheduling, the intersection manager sends responses to the UAVs that sent requests. A response to a UAV includes a UAV's scheduled entrance time to the intersection and the path of the UAV in the intersection. The scheduling and responses to all scheduled UAVs must be completed within $k\Delta t$.

A UAV in the approaching area is expected to follow the following rules:

- It cannot change lanes
- It cannot overtake a UAV ahead
- It must travel at constant altitude
- It must maintain its speed within $[s_{min}, s_{max}]$, unless it needs to slow down to avoid collision with UAV ahead or delay its entrance to the intersection. We will discuss in Section 4 how a UAV will decide its speed such that it can avoid collision with a UAV ahead and enter the intersection as scheduled.
- It must send reservation request to the intersection manager as soon as it enters the reservation zone. A reservation request message contains the UAV ID, time the request was sent, UAV's position when it sent the request, UAV's lane, and UAV's size.
- It enters the acceleration zone such that it can enter the intersection as scheduled.

Details on the length of the zones in the approaching area are described as follows:

- Reservation zone

A UAV must send its reservation request to the intersection manager as soon as it enters this zone. After it has sent its request, it should receive its schedule before it enters the next zone. Since the intersection manager processes the requests every $k\Delta t$, up to one $k\Delta t$ can be spent before its request can be processed and another $k\Delta t$ is needed for the scheduling. Hence, it will wait up to $2k\Delta t$ before it receives its schedule. If the UAV moves at the fastest speed possible, s_{max}, then it will travel up to $2k\Delta t s_{max}$ before it receives its request. Thus, to make sure that a UAV traveling at s_{max} will receive its schedule before it enters the next zone, the length of the reservation zone, l_{rz}, should be at least

$$l_{rz} \geq 2k\Delta t s_{max} \tag{1}$$

- Queueing zone

After the reservation zone, UAVs will enter the queueing zone. It is expected that a UAV has already been scheduled by the time it enters this zone. In this zone, it must adjust its speed to be able to enter the intersection as scheduled. It will either maintain its speed, speed up, slow down, or stop and wait for some time before entering the next zone, the acceleration zone.

Assume a UAV entered this zone at the speed of s_{max}, there is no UAV ahead, and it must stop for some time before it enters the next zone to be able to follow its schedule. Then, it will travel a distance of

$$d_{stop} = \frac{{s_{max}}^2}{-2\, r_{min}} \tag{2}$$

where r_{min} is the fastest possible negative acceleration rate of all UAVs. Thus, to make sure that the UAV can stop before it reaches the entrance of the acceleration zone, the length of the queueing zone, l_{qz} must be at least

$$l_{qz} \geq \frac{{s_{max}}^2}{-2\, r_{min}} \tag{3}$$

In case a UAV did not receive a response by the time it enters the queueing zone, then it is assumed that the message is lost. When this happens, it will resend a new request message while slowing down to stop at entrance of the acceleration zone. Then, it waits for its new schedule and will enter the acceleration zone as scheduled. Since this UAV causes the UAVs behind it to stop, the UAVs behind must also send requests again.

- Acceleration zone

A UAV can enter the acceleration zone only when it can enter the intersection at the scheduled time. If a UAV's speed is not in the allowed speed range when it enters this zone, then it must adjust its speed until it reaches the allowed minimum speed in the intersection.

Assume a UAV stopped before the entrance of the acceleration zone, then to be able to reach the speed requirement before it enters the intersection, the length of the acceleration zone, l_{az}, must be at least

$$l_{az} \geq \frac{{s_{max}}^2}{2\, r_{max}} \tag{4}$$

where r_{max} is the fastest possible positive acceleration rate of all UAVs.

4. UAV's Behavior in the Approaching Area

Consider a UAV, UAV_B, and a UAV ahead of UAV_B, which is UAV_A. In this section, we discuss how a UAV_B should control its speed while travelling in the approaching area such that it can avoid a collision with a UAV_A and such that it can enter the intersection at the scheduled time. This depends on the locations of both UAV_B and UAV_B in the approaching area. The different scenarios to be considered based on their locations and how UAV_B should behave in each scenario are listed below. Table 1 lists the symbols used in this section.

(1) UAV_B is in the reservation zone

When UAV_B is in the reservation zone, it may or may not have been scheduled yet to enter the intersection because the latest time it can be scheduled is when it reaches the queueing zone. Hence, when UAV_B is in the queueing zone, it just needs to avoid collision with a UAV ahead, if there is any. In case there is no UAV ahead, then UAV_B maintains its speed.

Listed below are different scenarios on how to avoid collision with UAV_A depending on its location.

(a) UAV_A is in the reservation zone or queueing zone

In this case, UAV_B must avoid collision with UAV_A. How to avoid collision with a UAV ahead will be discussed in Section 4.1.

(b) UAV_A is in the acceleration zone

Since UAV_B is still in the reservation zone, it has enough distance from UAV_A. Hence, UAV_B maintains its speed until it reaches the queueing zone.

(2) UAV_B is in the queueing zone

(a) UAV_{A} is in the queueing zone

Since UAV_{A} is still in the queueing zone, UAV_B must avoid collision with UAV_A, which will be discussed in Section 4.1.

(b) UAV_{A} is in the acceleration zone

UAV_B is in the queueing zone, which means it was already scheduled. There is no UAV ahead in the queueing zone, hence it must adjust its speed such that it can enter the intersection as scheduled. How to enter the intersection as scheduled will be discussed in Section 4.2.

(3) UAV_B is in acceleration zone

In the acceleration zone, UAV_B will accelerate at the rate of r_{max} until its speed reaches s_{max} . Afterwards, it must maintain its speed.

Table 1. List of symbols used in this section.

Symbol	Definition
Δt	Time is discretized every Δt
r_{min}	Fastest possible negative deceleration rate of all UAVs
r_{max}	Fastest possible positive acceleration rate of all UAVs
d_{min}	Minimum allowed distance between two UAVs
dr_{max}	Absolute value of r_{min}
$s_{B,j}$	Speed of UAV_B at t_j
$r_{B,j}$	Acceleration/deceleration rate of UAV_B from t_j to t_{j+1}
$d_{B,j}$	Distance UAV_B travels from t_j to t_{j+1}
$d_{A,B,j}$	Distance between UAV_A and UAV_B at t_j.
$d_{A,B}^{stop}$	Distance between UAV_A and UAV_B when both have stopped in the lane
$r_{B,j+1}^{d_{min}}$	Value of $r_{B,j+1}$ such that $d_{A,B}^{stop}$ is equal to d_{min}
$r_{B,j+1}^{s_{max}}$	Value of $r_{B,j+1}$ such that $s_{B,j+2}$ is equal to s_{max}
$d_{A,j+1}^{stop}$	Distance UAV_A travels from t_{j+1} until it stops
$t_{B,j}^{sched}$	Time UAV_B is scheduled to enter the intersection from t_j
$t_{B,j}^{inter}$	Time UAV_B spends from t_j until it enters the intersection
$r_{B,j+1}^{qz}$	Constant acceleration/deceleration rate UAV_B uses from t_{j+1} until it reaches the exit of the queueing zone
$t_{B,j}^{qz}$	Time UAV_B spends in the queueing zone from t_j
t_B^{wait}	Time UAV_B spends waiting at the entrance of the acceleration zone
t_B^{az}	Time UAV_B spends in the acceleration zone
$r_{B,j+1}^{stop}$	Deceleration rate UAV_B uses from t_{j+1} such that its speed at the entrance of the acceleration zone equal to zero
$t_{B,j}^{*inter}$	Value of $t_{B,j}^{inter}$ when t_B^{wait} is set to zero and $r_{B,j+1}^{qz}$ is set to $r_{B,j+1}^{stop}$
s_B^{azen}	Speed of UAV_B when it reaches the entrance of the acceleration zone
t_B^{acc}	Time UAV_B spends to accelerate from s_B^{azen} to s_{max} in the acceleration zone
d_B^{acc}	Distance UAV_B travels while accelerating from s_B^{azen} to s_{max}.
$t_B^{maintain}$	Time UAV_B spends to reach the exit of the acceleration zone while maintaining its speed after reaching s_{max}

4.1. Avoiding Collision with a UAV Ahead

For UAV_B to avoid collision with UAV_A, UAV_B should control its speed such that its distance from UAV_A is always greater than or equal to the minimum allowed distance between UAVs, d_{min}. To control its speed, UAV_B determines $r_{B,j+1}$ at t_j such that its distance from UAV_A is always at least d_{min}, where $r_{min} \leq r_{B,j+1} \leq r_{max}$.

In order to determine $r_{B,j+1}$ such that the distance between UAV_B and UAV_A is always at least d_{min} until UAV_A enters the intersection, UAV_B should know the rates that UAV_A will use from t_{j+1} until it enters the intersection. However, UAV_B cannot obtain this information since UAV_A determines the acceleration rate it will use only one Δt in

advance, just like UAV_B. This means, UAV_A determines $r_{A,j+1}$ at t_j. Hence, at t_j, UAV_B only knows $r_{A,j-1}$.

To address this problem, we assume that UAV_A decelerates at the fastest possible rate, dr_{max}, from t_{j+1} until its stops. We assume this case since the distance between them at any time in the future will be greater than d_{min} when UAV_A either accelerates or decelerates at a rate smaller than dr_{max}.

Now we will explain how to determine $r_{B,j+1}$. At t_j, the available parameters are $s_{A,j}$, $s_{B,j}$, $d_{A,B,j}$, $r_{A,j}$, $r_{B,j}$. From those values, we can get $s_{A,j+1}$, $s_{B,j+1}$, and $d_{A,B,j+1}$. As an example, consider the values of $r_{A,j}$, $r_{B,j}$, and $d_{A,B,j+1}$ shown in Figure 2 and set d_{min} as 1 m. The blue dashed lines represent the distance travelled by UAV_A from t_{j+1} assuming it will slow down using dr_{max} until it stops. A dashed line in the figure means the UAV decelerates at dr_{max}. Since we assume that UAV_A will slow down at dr_{max} until it stops, UAV_B should also slow down until it stops to avoid collision with UAV_A. One approach is to make UAV_B slow down using dr_{max} from t_{j+1} until it stops, as shown by the yellow dashed lines in the figure. While this makes the distance between them always at least d_{min}, we do not want UAV_B to slow down at rate dr_{max} when it can use a smaller deceleration rate that will still maintain the minimum distance from UAV_A. In this approach, as can be seen from the figure, the shortest distance between them, which is when both UAVs have stopped, is twice the value of d_{min}. This means UAV_B can use a smaller deceleration rate than dr_{max} and still maintain d_{min}. Moreover, the distance between them will be greater than the value in the example when UAV_A does not actually slow down using dr_{max} from t_{j+1}. Hence, to avoid this scenario, we assume that UAV_B will slow down using dr_{max} at t_{j+2}, as shown by the red dashed lines. Then, we get $r^{d_{min}}_{B,j+1}$, the value of $r_{B,j+1}$ such that the distance when both have stopped, $d^{stop}_{A,B}$, is equal to d_{min}.

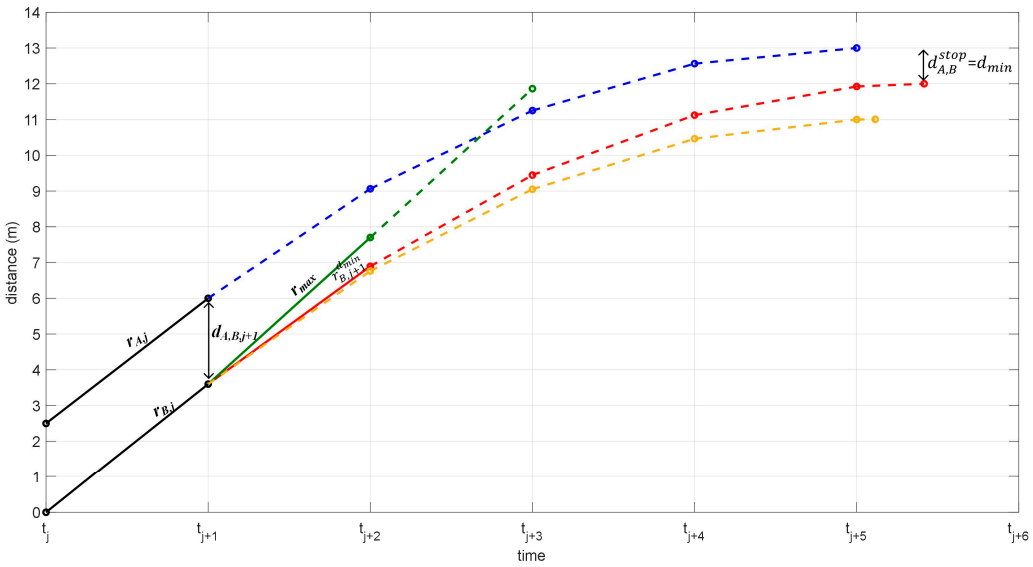


Figure 2. Distance between UAV_A and UAV_B on different acceleration rates. The lines represent the acceleration rates of UAV_A and UAV_B at different time steps. A dashed line in the figure means the UAV decelerates at dr_{max}. The blue lines represent the acceleration rates of UAV_A, slowing down from t_{j+1} until it stops. The yellow lines represent the acceleration rates of UAV_B, slowing down from t_{j+1} until it stops. The red lines represent the acceleration rates of UAV_B, slowing down starting at t_{j+2} until it stops. The value of $d^{stop}_{A,B}$ is equal to d_{min} when UAV_A and UAV_B follow the acceleration rates represented by the blue and yellow lines respectively. The green lines represent the acceleration rate of UAV_B when $r_{B,j+1}$ is equal to r_{max} and then it decelerates using dr_{max} at t_{j+2}. In this case, UAV_B collides with UAV_A before t_{j+3}.

We also show another scenario, as shown by the green solid line, wherein the distance travelled by UAV_B when $r_{B,j+1}$ is equal to r_{max} to show that there can be collision even when $d_{A,B,j+2}$ is greater than d_{min}. In this scenario, even though $d_{A,B,j+2}$ is greater than d_{min}, UAV_B will collide with UAV_A before t_{j+3} even when UAV_B will decelerate using dr_{max} at t_{j+2}. This happens when $s_{B,j+2}$ is greater than $s_{A,j+2}$ because this makes $d_{B,j+2}$ greater than $d_{A,j+2}$. Hence, we must make sure that distance between them is always at least d_{min} until they both have stopped.

Therefore, UAV_B should use a rate less than or equal to $r_{B,j+1}^{d_{min}}$ for the distance between them to be always at least d_{min}. However, UAV_B cannot just use any rate less than or equal to $r_{B,j+1}^{d_{min}}$. The speed of UAV_B at t_{j+2} should not exceed s_{max} and the rate must not exceed r_{max} . Considering this, let $r_{B,j+1}^{s_{max}}$ be the rate that will make $s_{B,j+2}$ equal to s_{max}. Then, the rate that UAV_B will use at t_{j+1} is

$$r_{B,j+1} = \min\left(r_{B,j+1}^{s_{max}},\ r_{B,j+1}^{d_{min}},\ r_{max}\right) \tag{5}$$

Note that this is to be done by UAV_B every Δt as long as there is a UAV ahead. Hence, if there is still UAV ahead at the next discrete time instant, then UAV_B determines $r_{B,j+1}$ again using the same approach and the new values of $s_{A,j}$, $s_{B,j}$, $d_{A,B,j}$, $r_{A,j}$, and $r_{B,j}$.

We now show how to determine $r_{B,j+1}^{d_{min}}$. Let $d_{A,j+1}^{stop}$ be the distance UAV_A will travel from t_{j+1} until it stops and let $d_{B,j+2}^{stop}$ be the distance UAV_B will travel from t_{j+2} until it stops. Then,

$$d_{A,B}^{stop} = d_{A,B,j+1} + d_{A,j+1}^{stop} - (d_{B,j+1} + d_{B,j+2}^{stop}). \tag{6}$$

From the equations of velocity, the values of $d_{A,j}$ and $d_{B,j}$ are obtained by

$$d_{U,j} = s_{U,j}\Delta t + \frac{r_{U,j}\ \Delta t^{\,2}}{2} \tag{7}$$

From Equations (6) and (7),

$$d_{min} = d_{A,B,j+1} + d_{A,j+1}^{stop} - s_{B,j+1}\Delta t - \frac{r_{B,j+1}^{d_{min}}\ \Delta t^{\,2}}{2} - d_{B,j+2}^{stop} \tag{8}$$

$$r_{B,j+1}^{d_{min}} = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \tag{9}$$

where

$$a = \Delta t^2, \tag{10}$$

$$b = \Delta t^2 a r_{min} + 2 s_{B,j+1}\Delta t, \tag{11}$$

$$c = 2dr_{max}\Big(d_{A,B,j+1} + d_{A,j+1}^{stop} - d_{min} - s_{B,j+1}\Delta t) + {s_{B,j+1}}^2. \tag{12}$$

4.2. Entering the Intersection as Scheduled

When UAV_B is in the queueing zone and there is no UAV ahead in the queueing zone, UAV_B must adjust its speed such that it can enter the intersection at the scheduled time. The time it will spend from t_j until it enters the intersection, $t_{B,j}^{inter}$, must be equal to $t_{B,j}^{sched}$, the time it is scheduled to enter the intersection from t_j. The value of $t_{B,j}^{inter}$ depends on the speed of UAV_B in the queueing zone and acceleration zone. In short, UAV_B uses a constant rate from t_{j+1} until it reaches the exit of the queueing zone, $r_{B,j+1}^{qz}$ where $r_{min} \leq r_{B,j+1}^{qz} \leq r_{max}$. Afterwards, it may spend some time waiting at the entrance of the acceleration zone. After it enters the acceleration zone, it accelerates at rate r_{max} until its speed reaches s_{max} . Then, it maintains its speed at s_{max} until it reaches the intersection. To be more specific, the

different scenarios on how UAV_B will behave at the queueing zone and acceleration zone based on $r^{qz}_{B,j+1}$ are listed below.

- Scenarios for which $r^{qz}_{B,j+1}$ is less than zero
  - Scenario 1: UAV_B slows down at rate $r^{qz}_{B,j+1}$ from t_{j+1} until it reaches the entrance of the acceleration zone. It reaches the entrance of acceleration zone with speed equal to zero. Then, it waits for some time before it enters the acceleration zone. In the acceleration zone, it accelerates at rate r_{max} until its speed reaches s_{max} . Then, it maintains its speed until it enters the intersection.
  - Scenario 2: UAV_B slows down at rate $r^{qz}_{B,j+1}$ from t_{j+1} until it reaches the entrance of the acceleration zone with speed equal to zero. Then, it immediately enters the acceleration zone and then accelerates at rate r_{max} until its speed reaches s_{max} . Then, it maintains its speed until it enters the intersection.
  - Scenario 3: UAV_B slows down at rate $r^{qz}_{B,j+1}$ from t_{j+1} until it reaches the entrance of the acceleration zone. It reaches the entrance of acceleration zone with speed greater than zero. Then, it enters the acceleration zone and accelerates at rate r_{max} until its speed reaches s_{max} . Then, it maintains its speed until it enters the intersection.
- Scenarios for which $r^{qz}_{B,j+1}$ is equal to zero
  - Scenario 4: UAV_B maintains its speed in the queueing zone, which is less than s_{max} . Then, it enters the acceleration zone and accelerates at rate r_{max} until its speed reaches s_{max} . Then, it maintains its speed until it enters the intersection.
  - Scenario 5: UAV_B maintains its speed in the queueing zone, which is equal to s_{max} . Then, it enters the acceleration zone and maintains its speed until it enters the intersection.
- Scenarios for which $r^{qz}_{B,j+1}$ is greater than zero
  - Scenario 6: UAV_B accelerates at rate $r^{qz}_{B,j+1}$ from t_{j+1} until it reaches the entrance of the acceleration zone. It reaches the entrance of the acceleration zone with speed less than s_{max} . In the acceleration zone, it accelerates at rate r_{max} until its speed reaches s_{max} . Then, it maintains its speed until it enters the intersection.
  - Scenario 7: UAV_B accelerates in the queueing zone at rate $r^{qz}_{B,j+1}$ from t_{j+1} until it reaches the entrance of the acceleration zone with speed equal to s_{max} . Then, it maintains its speed until it enters the intersection.

Let $t^{qz}_{B,j}$ be the time UAV_B will spend in the queueing zone from t_j. Let t^{wait}_B be the time it will spend waiting at the entrance of the acceleration zone. Let t^{az}_B be the time it will spend in the acceleration zone. Thus,

$$t^{inter}_{B,j} = t^{qz}_{B,j} + t^{wait}_B + t^{az}_B. \tag{13}$$

Note that t^{wait}_B can be greater than zero only when its speed at the entrance of the acceleration zone is zero, as described in Scenario 1. For UAV_B to enter the intersection as scheduled, UAV_B needs to determine t^{wait}_B and $r^{qz}_{B,j+1}$ such that $t^{sched}_{B,j}$ is equal to $t^{inter}_{B,j}$. For this, we first determine if UAV_B should slow down until it stops at the entrance of the acceleration zone as described in Scenario 1 and Scenario 2. To do this, we get $t^{*inter}_{B,j}$, the value of $t^{inter}_{B,j}$ when $t^{wait}_B = 0$ and $r^{qz}_{B,j+1} = r^{stop}_{B,j+1}$, where $r^{stop}_{B,j+1}$ is the negative acceleration rate that will make the speed of UAV_B at the entrance of the acceleration zone equal to zero.

In the queueing zone, UAV_B will change its speed from $s_{B,j+1}$ to s^{azen}_B, the speed of UAV_B when it reaches the entrance of the acceleration zone. Hence,

$$t_{B,j}^{qz} = \Delta t + \frac{s_B^{azen} - s_{B,j+1}}{r_{B,j+1}^{qz}}. \tag{14}$$

Let t_B^{acc} be the time it will take to accelerate from s_B^{azen} to s_{max} in the acceleration zone. Let d_B^{acc} be the distance it will travel while accelerating from s_B^{azen} to s_{max}. Let $t_B^{maintain}$ be the time it will take to reach the exit of the acceleration zone while maintaining its speed after reaching s_{max}. Then,

$$t_B^{acc} = \frac{s_{max} - s_B^{azen}}{r_{max}} \tag{15}$$

$$d_B^{acc} = \frac{{s_{max}}^2 - s_B^{azen2}}{2r_{max}} \tag{16}$$

$$t_B^{maintain} = \frac{l_{az} - d_B^{acc}}{s_{max}} \tag{17}$$

$$t_B^{az} = t_B^{acc} + t_B^{maintain}. \tag{18}$$

We now show how to determine $r_{B,j+1}^{qz}$ and t_B^{wait} based on the value of $t_{B,j}^{*inter}$. For this we consider three cases as listed below.

(1) $t_{B,j}^{*inter} < t_{B,j}^{sched}$

If $t_{B,j}^{*inter} < t_{B,j}^{sched}$, then this means UAV_B will enter the intersection earlier than scheduled when it will slow down until stop and then immediately enters the acceleration zone. Thus, it must spend some time at the entrance of the acceleration zone, as described in Scenario 1. Hence, $r_{B,j+1}^{qz} = r_{B,j+1}^{stop}$ and t_B^{wait} is obtained by

$$t_B^{wait} = t_{B,j}^{sched} - t_{B,j}^{qz} - t_B^{az}. \tag{19}$$

(2) $t_{B,j}^{*inter} = t_{B,j}^{sched}$

If $t_{B,j}^{*inter} = t_{B,j}^{sched}$, then it will enter the intersection as scheduled when it will use rate of $r_{B,j+1}^{stop}$ and enter the acceleration zone immediately, as described in Scenario 2. Hence, $r_{B,j+1}^{qz} = r_{B,j+1}^{stop}$ and $t_B^{wait} = 0$.

(3) $t_{B,j}^{*inter} > t_{B,j}^{sched}$

If $t_{B,j}^{*inter} > t_{B,j}^{sched}$, then this means UAV_B will enter the intersection at a later time than scheduled when it will use a rate of $r_{B,j+1}^{stop}$. Hence, UAV_B must enter the acceleration zone at an earlier time and should use a rate higher than $r_{B,j+1}^{stop}$. Using a rate higher than $r_{B,j+1}^{stop}$ will make the speed of UAV_B at the entrance of the acceleration zone greater than zero, thus $t_B^{wait} = 0$. From Equations (13) and (14), the value of $r_{B,j+1}^{qz}$ that will make $t_{B,j}^{sched}$ equal to $t_{B,j}^{inter}$ is

$$r_{B,j+1}^{qz} = \frac{s_B^{azen} - s_{B,j+1}}{t_{B,j}^{sched} - t_B^{az} - \Delta t}. \tag{20}$$

If $r_{B,j+1}^{qz} < 0$, then UAV_B will behave as described in Scenario 3. If $r_{B,j+1}^{qz} = 0$, then it will behave as described in Scenario 4 when $s_{B,j+1} < s_{max}$, while it will behave as described in Scenario 5 when $s_{B,j+1} = s_{max}$. If $r_{B,j+1}^{qz} > 0$, then it will behave as described in Scenario 6 when $s_B^{azen} < s_{max}$, while it will behave as described in Scenario 7 when $s_B^{azen} = s_{max}$.

5. Paths of a UAV in the Intersection

We define the allowed movements in the intersection to reduce the avoidable complexity in the trajectories of the UAVs. Figure 3 shows the UAVs' allowed paths in the intersection in two-dimensional (2D) view based on their entrance lanes. If a UAV is to turn left, it should enter the intersection only through the two left-most entrance lanes. If it is to move straight, it will take the middle or second to the right-most lane. If it is to turn right, it will only take the right-most lane. UAVs can only enter and exit the intersection through the middle layer. This rule is applied to all the entrance ways.

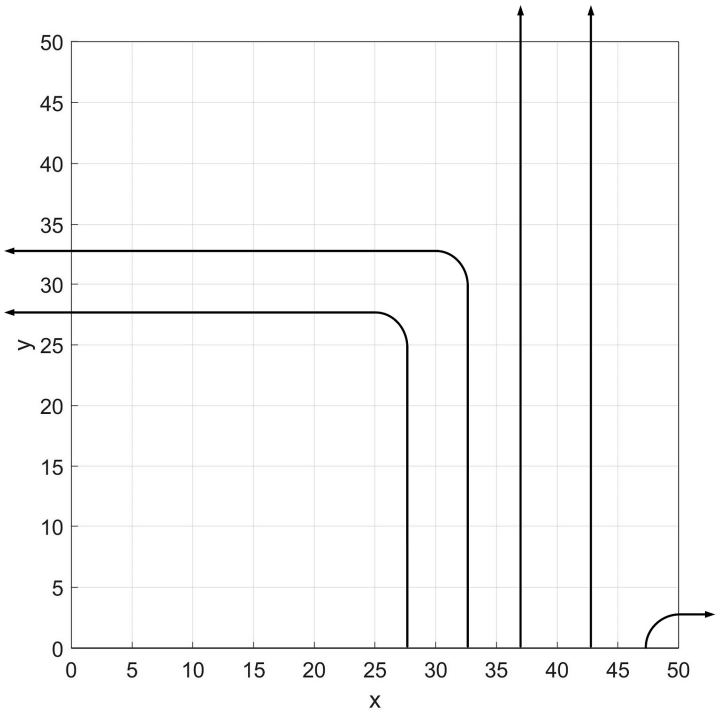


Figure 3. Allowed paths of UAVs based on entrance lane.

The allowed movements in the 3D intersection are shown in Figure 4. The allowed movements are left turn, move forward, right turn, move downward then turn left, move downward then forward, move upward then turn left, and move upward then forward. The movements in Figure 4 are for UAVs entering the south way. Similar movements are to be followed by UAVs entering from other ways.

From the allowed paths shown in Figure 3, some movements will not be allowed for a UAV based on its entrance lane. For example, if a UAV is at the left most lane, then it cannot turn right in the intersection. Also, note that a UAV that will turn right can only enter the intersection from the right most lane, thus it will exit the intersection immediately and it can only move as shown in Figure 4c.

To define the positions of the UAVs while moving in the intersection, the intersection is divided into blocks, and we define one waypoint at the center of each side of every block. The waypoints of the blocks are the black circles in the Figure 4. A block's length and width are equal to the lane's width while its height is equal to the layer's height. For simplicity, we assume a block has equal dimensions. When a UAV moves in the intersection, a movement should start with the UAV's center at the waypoint of a block and end with the UAV's center at the waypoint of the next block in the path. However, there is an exception that is, when the block is connected to the exit lane, then the UAV will not enter another block, but instead it will exit the intersection. For example, if a UAV is to turn left, as shown in Figure 4a, it will enter the waypoint at the back side of the current block. Then, it will turn to the left while maintaining its altitude until it reaches the waypoint at the right side of the next block in the path.

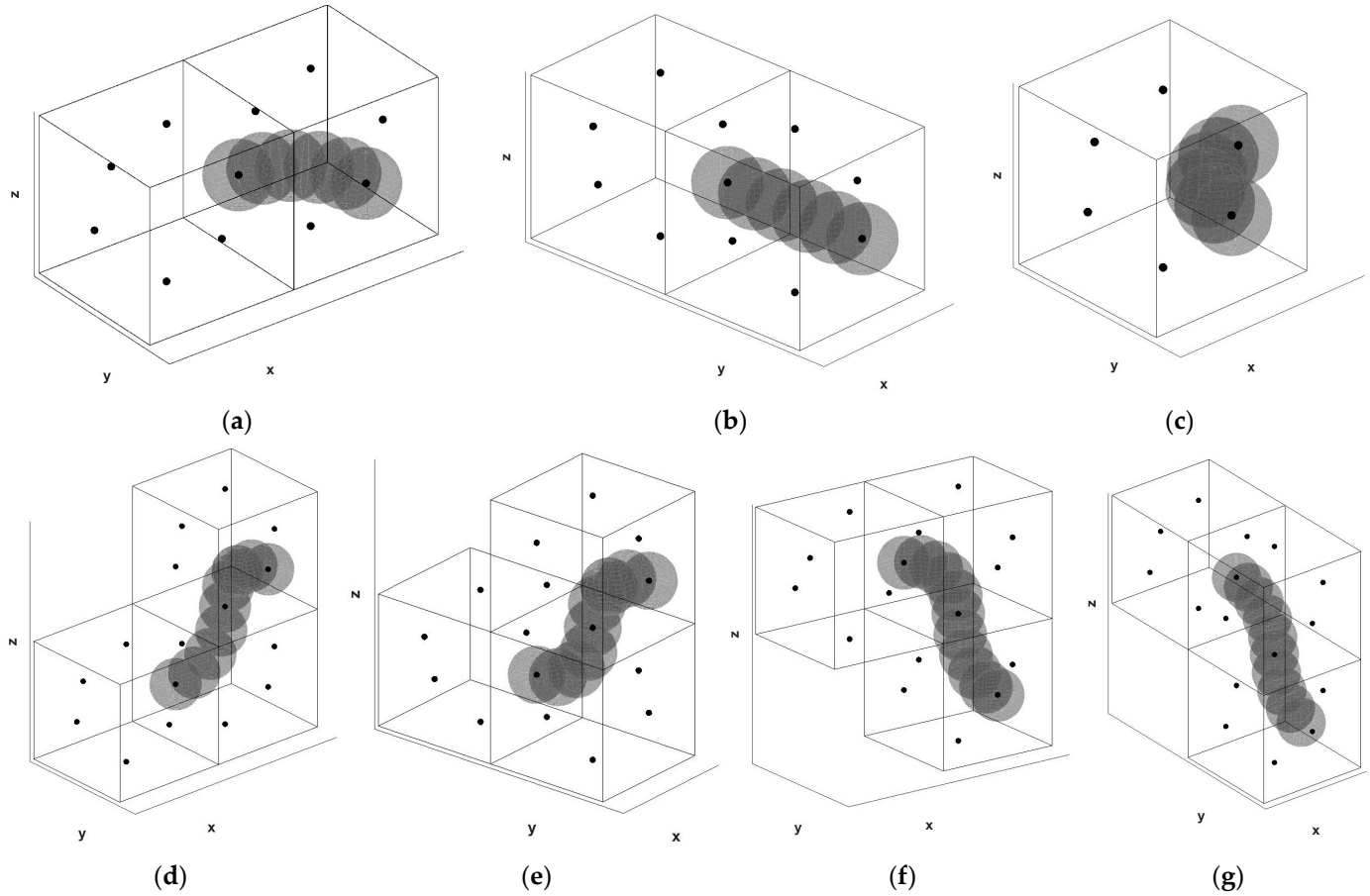


Figure 4. Possible movements of a UAV in the intersection (**a**) Turn left on the same layer (**b**) Move forward on the same layer (**c**) Turn right on the same layer (**d**) Move down then turn left (**e**) Move down then forward (**f**) Move up then turn left (**g**) Move up then forward.

The turning movements in Figure 4a,c follow the arc of a circle with radius equal to the half of the block's width and angle of 90 degrees. The turning movements in Figure 4d–g follow the arc of a circle with radius equal to the half of the height of the block and angle of 90 degrees.

6. Scheduling Scheme

Scheduling a UAV means planning for a UAV's path and entrance time in the intersection. There are multiple possible paths that a UAV can take since the intersection is multi-layered. For that reason, the goal is to find a path such that the UAV can exit the intersection the fastest.

In path finding, a graph represents the environment where the path is searched. A graph consists of a set of nodes and edges that connect them, while a path is a sequence of nodes from the start node to the end node.

In our scheduling scheme, we find a path in a graph where the nodes are the waypoints of the blocks, and the edges are the movements of the UAV in the intersection. For example, when a UAV is to move left and will enter the left-most lane, the graph where a path is to be searched is shown in Figure 5. The black circles represent the nodes, the blue lines represent the edge from a node to another node in a lower layer, the red lines represent the edge from one node to a node in a higher layer, and the green lines represent the edge from one node to another node in the same layer. The start node is the node at the entrance of the intersection, while the end node is the node at the exit of the intersection. To show the graph more closely, we show its y-z view in Figure 6.

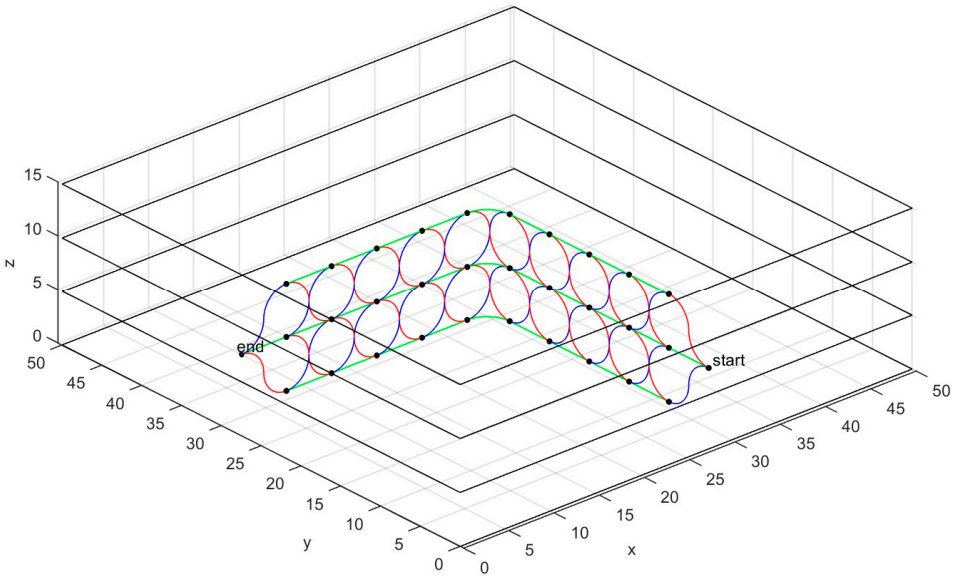


Figure 5. 3D view of graph to be searched for a UAV entering from the left-most entrance lane.

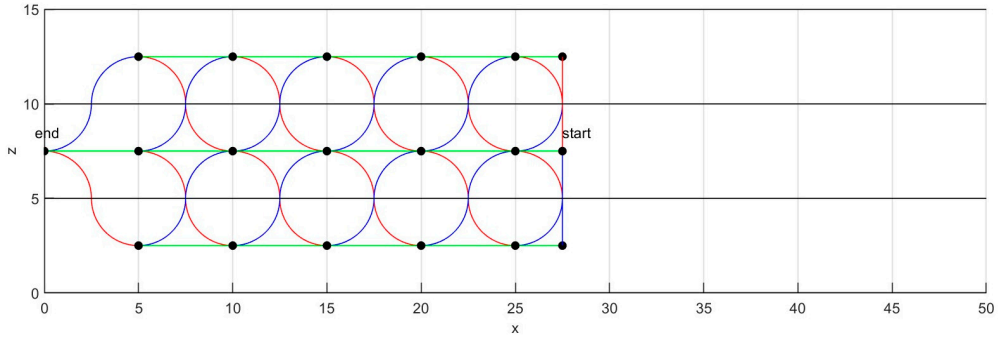


Figure 6. y-z view of Figure 5.

Along with path finding, we implement the reservation-based scheduling approach to allow multiple UAVs to be scheduled in the intersection at the same time without collision. In this approach, multiple UAVs can be in the intersection without collision by reserving a space in the intersection to one UAV at a time. Thus, we divide the intersection into cubes to discretize the space in the intersection that will be reserved by the UAVs. Note that in this paper, the cubes are not the same with the blocks and a block can contain multiple cubes.

To explain how reservation-based scheduling is implemented along with pathfinding, suppose we are finding a path for a UAV, let $node_n$ be the node currently visited and let $node_s$ be the successor node of $node_n$. A successor node of $node_n$ is a node that can be directly visited from $node_n$. Before $node_s$ can be included in the path finding, the cubes that the UAV will pass through when it will travel from $node_n$ to $node_s$ must not be reserved by other UAVs. We will discuss the scheduling scheme in detail in Section 6.3.

To keep track of which cubes are reserved by which UAVs and at the times they are reserved, we use red–black trees [56] to store the reservations at each cube. Each cube has a red-black tree that contains the reservations on the cube. An example of a red–black tree is shown in Figure 7. One node of the red–black tree represents a reservation of a UAV. A node contains the reserved arrival and departure times of a UAV on that cube with respect to the current simulation time.

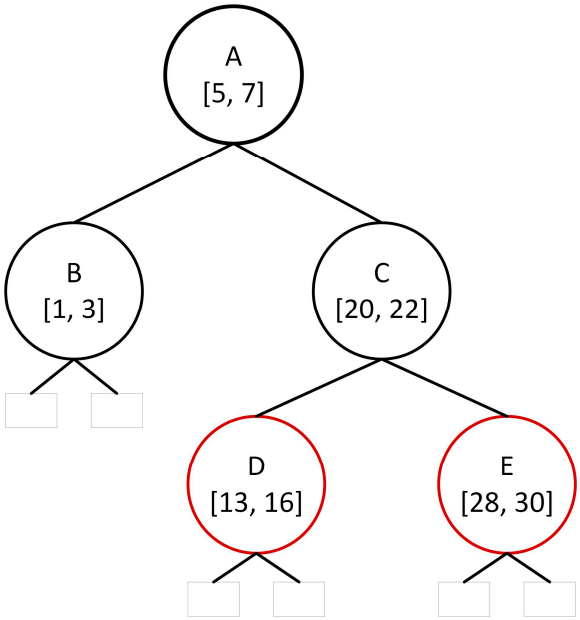


Figure 7. Example red–black tree of a cube.

If there are multiple UAVs to be scheduled, the UAVs are scheduled in sequence. It is important to note that the sequence of the UAVs has an effect on the UAVs' schedules. For example, we schedule $\{UAV_1, UAV_2, UAV_3\}$ in order. Since UAV_1 is scheduled first, the available cubes that UAV_2 can reserve depends on the reserved cubes by UAV_1. Likewise, the available cubes to UAV_3 depends on the reservations of UAV_1 and UAV_2. Therefore, we implement genetic algorithm to decide the scheduling sequence that will be used. The implementation of genetic algorithm will be discussed in the next subsection.

6.1. Using Genetic Algorithm to Decide the Scheduling Sequence

Genetic algorithm is used to decide the sequence of UAVs in scheduling. To apply the genetic algorithm, one UAV is represented as a gene and one chromosome represents a sequence of the UAVs in scheduling. Hence, the size of a chromosome is equal to the number of UAVs to be scheduled.

For the first generation, we generate C chromosomes to create different sequences of UAVs. We have a condition when generating a chromosome that for UAVs that are in the same lane, the UAV ahead should appear first in the chromosome so that it will be scheduled first. Then, the UAVs are temporarily scheduled according to their order in the chromosome. We will discuss how we schedule a UAV in Section 6.3. We temporarily schedule the UAVs since the final schedule will be based on the best chromosome. After scheduling all UAVs in a chromosome, we compute the chromosome's fitness value based on the objective function. The objective is to minimize

$$\sum_{U=1}^{m} t_{travel_U} \tag{21}$$

where t_{travel_U} is the travel time of UAV_U from the time it sent request until it exits the intersection and m is the total number of UAVs scheduled in the current scheduling instant.

Then, we select the chromosomes with the top fifty percent highest fitness values. From the chosen top chromosomes, we perform crossovers and mutations to generate new chromosomes and complete the population of C chromosomes for the next generation. These processes are performed repeatedly to improve the fitness values of the chromosomes. The genetic algorithm is terminated when it reaches Gth generation. The final scheduling sequence of the UAVs will be based on the best chromosome of the Gth generation.

6.2. Tasks of the Intersection Manager

In summary, the tasks of the intersection manager at every scheduling epoch are to schedule the UAVs, to send the responses to the UAVs, and to update the times of the saved reservations for the next scheduling epoch. A response to a UAV includes its scheduled time to enter the intersection and the path in the intersection that it must follow.

The scheduling scheme implements a genetic algorithm to find the best scheduling sequence. After choosing the best sequence, the intersection manager finalizes the reservations for the best sequence by saving the reservations to the respective cubes' red–black trees. Then, the intersection manager sends responses to the UAVs. The stored reservation times are relative to the current scheduling epoch, thus to prepare for the next scheduling epoch, we update the stored reservation times by decreasing their values by $k\Delta t$. Also, a UAV's reservation to a cube will be deleted if the reserved departure time of the UAV is less than or equal to zero, which means the UAV has already passed through that cube. The summary of the scheduling done at every $k\Delta t$ is shown in Figure 8.

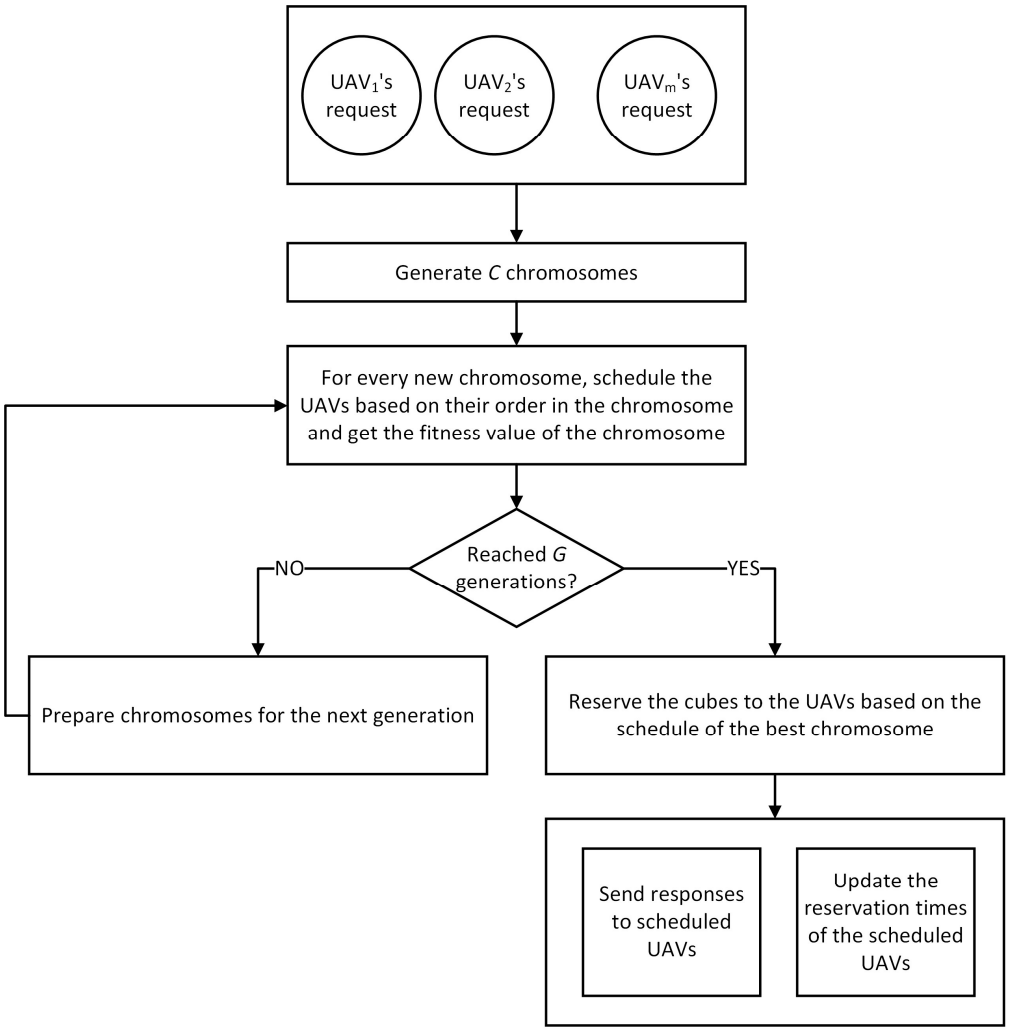


Figure 8. Summary of the tasks of the intersection manager every $k\Delta t$.

6.3. Scheduling a UAV

When scheduling a UAV, the intersection manager decides the time that the UAV will enter the intersection and the path of the UAV in the intersection. Denote t_e as the earliest time that the UAV can reach the entrance of the intersection, then the UAV can be scheduled to enter the intersection any time from t_e as long as its path in the intersection does not conflict with other UAVs. The paths without conflict depend on the time the UAV will enter

the intersection because the reserved cubes of the other UAVs change over time. Hence, to decide the time that the UAV will enter the intersection, we plan the path of the UAV on different intersection entrance times and choose the path with the fastest intersection exit time.

Specifically, we plan the path of the UAV in the intersection given the entrance time to the intersection t_{inter}, where $t_{inter}^{lb} \leq t_{inter} \leq t_{inter}^{ub}$. t_{inter}^{lb} is the earliest time that the UAV can enter the intersection and t_{inter}^{ub} is the latest time that the UAV can enter the intersection such that it is possible to find the fastest path in the intersection. If there is a UAV ahead, then it can enter the intersection only after the UAV ahead has entered the intersection. Otherwise, then it can enter the intersection at the time it will arrive at the entrance of the intersection given that it will not experience delay from the time it sent its request. Thus,

$$t_{inter}^{lb} = \max(t_a,\ t_e) \tag{22}$$

where t_a is the reserved departure time of the UAV ahead from the first cube it will occupy in the intersection and t_e is the earliest time that the UAV can reach the entrance of the intersection, assuming that it will not experience delay from the time it sent its request.

The pseudocode of the algorithm for scheduling a UAV is in Algorithm 1. In Algorithm 1, we initially set t_{inter} to t_{inter}^{lb}, set t_{inter}^{ub} to null and set path P as an empty set where P will contain the nodes of the path that the UAV will use. Then, we find a path given that the UAV will enter the intersection at t_{inter} through the function *findfastestpath*. Details on *findfastestpath* will be discussed in the next subsection. In the algorithm, we find a path iteratively until t_{inter} reaches the upper bound t_{inter}^{ub}. The value of t_{inter} is increased by Δt at each iteration. At each iteration, if a new path *newP* is found and the *newP*'s exit time is faster than the exit time of current P, set the *newP* as P and assign

$$t_{inter}^{ub} = t_{exit}^{P} - t_{ml}. \tag{23}$$

where t_{exit}^{P} is P's time to exit the intersection and t_{ml} is the travel time in the intersection when the path passes only across the middle layer. After finding the fastest path, we reserve the cubes that the UAV will occupy in P. A UAV is reserved to a cube by inserting an associated node to the red–black tree of the cube.

Algorithm 1 Scheduling a UAV

scheduleUAV (UAV ID, UAV's size, UAV's lane, time the request was sent)
Get t_{inter}^{lb}
$t_{inter} = t_{inter}^{lb}$
Set P as empty set
Set t_{inter}^{ub} as null
while t_{inter}^{ub} is null or $t_{inter} < t_{inter}^{ub}$
    $newP \leftarrow findfastestpath$ (t_{inter}, UAV's size, UAV's lane)
    if P is null or *newP*'s exit time < P's exit time **then**
        $P \leftarrow newP$
        $t_{inter}^{ub} = t_{exit}^{P} - t_{ml}$
    end if
    $t_{inter} = t_{inter} + \Delta t$
end while
reserve the cubes the UAV will occupy in P

To explain the value assigned to t_{inter}^{ub} in Equation (23), let $t_{inter,\,i}$ be the value of t_{inter} on the ith iteration of *findfastestpath*, while let $t_{path,\,i}$ and $t_{exit,\,i}$ be the travel time in the intersection and the exit time from the intersection of the path found in ith iteration, respectively. Then,

$$t_{exit,\,i} = t_{inter,\,i} + t_{path,\,i}. \tag{24}$$

At jth iteration where $j > i$, even though the UAV will enter the intersection at a later time than $t_{inter,\,i}$, the value of $t_{exit,\,j}$ will still be smaller than $t_{exit,\,i}$ when $t_{inter,\,j} + t_{path,\,j} < t_{exit,\,i}$, which is when $t_{inter,\,j} < t_{exit,\,i} - t_{path,\,j}$. In other words, assuming we found a faster path at the ith iteration and we set this path as P and set P's time to exit the intersection as t^P_{exit}, then a path found in the later iteration j is faster than t^P_{exit} only when $t_{inter,\,j} < t^P_{exit} - t_{path,\,j}$. We know that the path that passes only across the middle layer is the path with the fastest travel time in the intersection, hence the smallest possible value of $t_{path,\,j}$ is t_{ml}. Thus, we only need to find path while $t_{inter} < t^P_{exit} - t_{ml}$. For that reason, we set new value to t^{ub}_{inter} using Equation (23) every time a new faster path is found.

- Finding the fastest path in the intersection: *findfastestpath*

Given UAV's size, UAV's entrance lane, and t_{inter}, the function *findfastestpath,* finds the path that will make the UAV exit the intersection the fastest. To start the path finding, we define the graph where a path will be searched based on the UAV's entrance lane (e.g., Figure 5). Then from the graph, we find a path from the start node to the end node with the fastest travel time in the intersection given that the UAV will enter the intersection at t_{inter}.

The path finding is done using modified A* search. A* search is a path finding algorithm that finds the path from the start to the end node with the smallest cost (e.g., shortest time or shortest distance). Suppose $node_n$ is the currently visited node in A* search, the next node to be visited is the successor node, $node_s$, with the smallest cost

$$f(s) = g(s) + h(s) \tag{25}$$

where $g(s)$ is the length of the path from start node to $node_s$ and $h(s)$ is the Manhattan distance from $node_s$ to end node. In addition, there is an open list and a closed list in A* search to keep track of the visited nodes. The nodes that are yet to be visited are in the open list, while the nodes that were already visited are in the closed list. The nodes in the closed list will not be visited anymore.

In our path finding algorithm, even though there is an edge between $node_n$ and $node_s$, a path can be made from $node_n$ to $node_s$ only when the UAV can travel from $node_n$ to $node_s$ without conflict with other reservations. This depends on the time the UAV will arrive at $node_n$ which depends on t_{inter} and the previously visited nodes from start node to $node_n$. Also, in our approach, even if $node_n$ was already previously visited, it can still be visited later if the nodes in the path from the start node to $node_n$ are different. Hence, we do not need to maintain a closed list since previously visited nodes can still be visited again. Note that the complexity of our algorithm is worse than A*, since a node can be visited more than once. The complexity will be further discussed in Section 7.1. The similarity of our algorithm with the A* is that among the successor nodes of $node_n$, the next node to be visited is the node with minimum value of $f(s)$ as defined in Equation (25).

Now we will explain our path finding approach. First, we add the start node to the open list. While open list is not empty, set $node_n$ as the node from the open list with minimum $f(n)$. For each successor $node_s$ of $node_n$, we check if the path from $node_n$ to $node_s$ does not conflict with other UAVs by calling the function *checkPathtoSuccessor*. It will return true if UAV can go from $node_n$ to $node_s$ without conflict. Otherwise, it will return false. Details on *checkPathtoSuccessor* will be discussed in the next subsection. If there is no conflict, then the UAV can move from $node_n$ to $node_s$, so we assign values to $h(s)$, $g(s)$, $f(s)$, and set $node_n$ as the parent of $node_s$. Then, we add $node_s$ to the open list. The nodes are repeatedly visited in this way until the end node is reached or when the open list is empty, which means no path is found. If the end node is reached, we obtain the path from start node to end node by backtracking the parents of the nodes starting from the end node until start node. The returned value P is the set of nodes from start node to end node. In summary, the pseudocode of the path finding algorithm is in Algorithm 2.

Algorithm 2 Finding the fastest path from start node to end node given the UAV will enter the intersection at t_{inter}

findfastestpath(t_{inter}, UAV's size, UAV's lane)
open list = {*start node*}
while *open list* is not emtpy
  set $node_n$ as node from open list with minimum $f(n)$
  remove $node_n$ from *open list*
  for each $node_n$'s successor $node_s$
    $noConflict \leftarrow checkPathtoSuccessor\ (node_n,\ node_s\ , t_{inter}, g(n)\ ,$ UAV size)
    if *noConflict* is true **then**
      $g(s)$ = $g(n)$ + length of the path from $node_n$ to $node_s$
      $h(s)$ = manhattan distance from $node_s$ to end node
      $f(s)$ = $g(s)$ + $h(s)$
      set $node_n$ as parent of $node_s$
      if $node_s$ is end node **then** stop search
      else add $node_s$ to open list
      end if
    end if
  end for
end while
if end node is reached **then**
  get path P from start node to end node
  return P
end if

- Checking the path between nodes: *checkPathToSuccessor*

Before $node_s$ can be included in the path finding, we need to check if the path from currently visited $node_n$ to $node_s$ does not conflict with other reservations. This means all of the cubes that the UAV will pass through from $node_n$ to $node_s$ must not be reserved by other UAVs at the time the UAV being scheduled will occupy them. Let *rc* be the set of cubes that it will occupy when it travels from $node_n$ to $node_s$, let *rta* be the set containing the earliest possible arrival times to each cube in *rc* and let *rtd* be set containing the latest possible departure times from each cube in *rc*. How to determine the values of *rc*, *rta*, and *rtd* will be explained in the next subsection.

The times in *rta* and *rtd* are relative from the time the UAV is at $node_n$. Hence, the values in *rta* and *rtd* must be adjusted by adding the time the UAV will arrive at $node_n$. The time the UAV will arrive at $node_n$ is equal to the time it will enter the intersection, t_{inter}, plus the time it will spend traveling from the start node until $node_n$. The time it will spend traveling from the start node to $node_n$ depends on the length of the path from start node to $node_n$ and the speed of the UAV in the intersection. Recall that UAVs can travel at any speed within [s_{min}, s_{max}]. If the UAV moves at s_{max} in the intersection, then it will arrive and depart from $node_n$ at the earliest time possible while if it moves at s_{min}, then it will arrive and depart from $node_n$ at the latest time possible. To guarantee collision avoidance on any speed that UAV might take between the allowed speed range, we reserve the cube to the UAV from the earliest possible time it will arrive at the cube until the latest possible time it will depart the cube. Thus, we get $t_n^{s_{max}}$, the time the UAV will reach $node_n$ if it will travel at s_{max} in the intersection and we also get $t_n^{s_{min}}$ the time the UAV will reach $node_n$ if it will travel at s_{min} in the intersection. The value of $t_n^{s_{max}}$ is added to every value in *rta*, while is $t_n^{s_{min}}$ added to every value in *rtd*.

$$t_n^{s_{max}} = t_{inter} + \frac{g(n)}{s_{max}} \tag{26}$$

$$t_n^{s_{min}} = t_{inter} + \frac{g(n)}{s_{min}} \tag{27}$$

After updating the values in rta and rtd, we will now check the availability of the cubes. For each cube k in rc, we check rc_k's red–black tree if there are no other reservations that will conflict from rta_k to rtd_k where rta_k is the earliest possible arrival time to rc_k and rtd_k is the latest possible departure time from rc_k . Note that a conflict does not mean that there will be sure collision. However, to make sure there will be no collision on any speed the UAV might take within $[s_{min}, s_{max}]$, then there must be no conflict to all of the cubes in rc for the UAV to be able to travel from $node_n$ to $node_s$. If at least one cube has conflict, then we stop checking for the other cubes in rc.

To check for conflict at cube rc_k, the red–black tree of rc_k is traversed to check the arrival and departure times of the current reservations at rc_k. The current reservations' arrival and departure times must not overlap with $[rta_k\ ,\ rtd_k\ ]$. To explain this in detail, let T be the red–black tree of rc_k, let x be a node in T, let x_{ta} be the arrival time of node x, let x_{td} be the departure time of node x, let x_l be the left node x, and let x_r be the right node x. We traverse the nodes of T until there is conflict or until the end of the tree has been reached. First, we set x to the root of T. If $rtd_k \leq x_{ta}$, we go to the left of x hence $x = x_l$. However, if $rta_k \geq x_{td}$, we go to the right of x hence $x = x_r$. Otherwise, it means there is an overlap with $[rta_k\ ,\ rtd_k\ ]$, so the cube is not available. Algorithm 3 shows the pseudocode of *checkPathtoSuccessor* function.

Algorithm 3 Checking if the cubes the UAV will occupy from $node_n$ to $node_s$ do not conflict with other reservations

checkPathToSuccessor $(node_n, node_s, t_{inter}, g(n)$, UAV size)
$rc \leftarrow rc(node_n, node_s,$ UAV size)
$rta \leftarrow rta(node_n, node_s,$ UAV size)
$rtd \leftarrow rtd(node_n, node_s,$ UAV size)
compute $t_n^{s_{max}}$
compute $t_n^{s_{min}}$
$cellAvailable \leftarrow$ true
for each cube k of rc
  $rta_k = rta_k + t_n^{s_{max}}$
  $rtd_k = rtd_k + t_n^{s_{min}}$
  T = k's Red-Black Tree
  node x = root of T
  while $x! = null$ and $cellAvailable$ is true
    if $rtd_k \leq x_{ta}$ **then** $x = x_l$
    else if $rta_k \geq x_{td}$ **then** $x = x_r$
    else $cellAvailable \leftarrow$ false
    end if
  end while
  if $cellAvailable$ is false **then**
    break the for loop
  end if
end for
return $cellAvailable$

As an example, consider the red–black tree T shown in Figure 7 and consider a case where the cube is not available. The cube is not available when rta_k = 15 and rtd_k = 20 since it will conflict with the values in $node_D$. First, we set x to the root of T thus $x = node_A$. $rta_k > x_{td}$ thus $x = x_r = node_C$. Then $rtd_k = x_{ta}$ hence $x = x_l = node_D$. Then $rtd_k > x_{ta}$ and $rta_k < x_{td}$, which means there is an overlap between [15, 20] and [13, 16], thus the cube is not available. Another example in which the cube is available is when rta_k = 3 and rtd_k = 5. In this example, first, set $x = node_A$. $rtd_k = x_{ta}$ hence $x = x_l = node_B$. $rta_k = x_{td}$ hence $x = x_r$ = null, thus cube is available.

- Determining the cubes a UAV will occupy between nodes

We need to determine the cubes that a UAV will occupy when it travels from $node_n$ to $node_s$ and the time duration it will reserve them to know which cubes should be checked

and if there will be conflict with other reservations. To determine the cubes and the time the UAV will occupy them, we simulate the movement of the UAV from $node_n$ to $node_s$ every Δt and we get the cubes it will occupy.

Recall that UAVs can move at any speed within $[s_{min}, s_{max}]$. If UAV moves at s_{max}, then it will arrive and depart from the cubes at the earliest time possible. If it moves at s_{min}, then it will arrive and depart at the latest time possible. Thus, we reserve the cubes from the earliest possible arrival time until the latest departure time. Hence, the simulation is done twice. One simulation for the UAV moving at s_{max} to get the earliest arrival time, and another simulation for the UAV moving at s_{min} to get the latest departure time.

The smoothness of the UAV's movement in the simulation depends on the granularity of Δt . We get the change in position of the UAV every Δt, thus using smaller Δt, results to smoother movement of the UAV in the simulation. Consequently, we cannot capture the exact times on when a UAV arrives and departs at each cube. To deal with the accuracy error in the times the UAV will occupy the cubes, we subtract one Δt from the arrival time to the cube and add one Δt to the departure time from the cube. In addition, to make sure that all cubes that the UAV will pass through are obtained in the simulation when it travels at s_{max}, we assign an upper bound limit to Δt that is

$$\Delta t \ < \frac{\text{UAV } diameter}{s_{max}}. \tag{28}$$

We will now explain the implementation of the simulation. Let *rc* be the set containing the cubes that the UAV will occupy when it travels from $node_n$ to $node_s$. Let *rta* be the set containing the arrival times to each cube in *rc* and let *rtd* be the set containing the departure times from each cube in *rc*. Before the start of the simulation, *rc*, *rta*, *rtd* are initialized as empty sets, then we populate them as we do the simulation. We start the simulation with the UAV's center positioned at $node_n$ and set the simulation time t_s to zero. The UAV will move with distance of d_{sim} at every Δt where

$$d_{sim} = \begin{cases} \Delta t \ s_{max}, \text{ 1st } simulation \\ \Delta t \ s_{min}, \text{ 2nd } simulation \end{cases}. \tag{29}$$

The cubes that the UAV occupies at the current simulation time are the cubes that intersects the UAV. For each cube that the UAV occupies at the current simulation time that is not in *rc*, we do the following: insert the cube to *rc*, insert $t_s - \Delta t$ to *rta*, and insert $t_s + \Delta t$ to *rtd*. If the cube is already in *rc*, meaning the cube was already occupied in the previous time step, then we update its *rtd* to the current simulation time by updating its *rtd* to $t_s + \Delta t$. After getting the cubes it occupies at the current simulation time, we add Δt to t_s for the next time step. These steps are repeated, and the simulation ends when UAV's center has already reached $node_s$. The summary of the simulation algorithm is in Algorithm 4.

Algorithm 4 Simulation to determine the cubes a UAV will occupy when it travels from $node_n$ to $node_s$

```
getCubesToOccupy(node_n, node_s, UAV's size)
initialize rc as empty set
initialize rta as empty set
initialize rtd as empty set
set simulation iteration i = 1
while i ≤ 2
    t_s = 0
    if i is equal to 1 then d_sim = Δt s_max
    else d_sim = Δt s_min
    end if
    position the UAV at node_n
    while UAV has not yet reached node_s
        move UAV by d_sim
        get the cubes that UAV occupies at current position
        for each cube it occupies
                if cube is not in rc then
                    insert cube to rc
                    insert t_s − Δt to rta
                    insert t_s + Δt to rtd
                else set rtd of cube to t_s + Δt
            end if
        end for
        t_s = t_s + Δt
    end while
    i = i + 1
end while
store rc(node_n, node_s, UAV size), rta(node_n, node_s, UAV size),
rtd(node_n, node_s, UAVsize)
```

The simulations are performed offline and in advance for every UAV size and pair of adjacent nodes in the intersection. The values in *rc*, *rta*, and *rtd* for a given pair of nodes and UAV's size are stored so that when finding a path for a UAV, the intersection manager already knows the cubes the UAV will occupy given $node_n$, $node_s$, and the UAV's size.

7. Discussion of the Scheduling Algorithm

7.1. Complexity of the Scheduling Algorithm

In this section, we will discuss the overall complexity of the scheduling algorithm as shown in Algorithm 1. First, we will discuss the complexity of *findfastestpath,* which is called inside Algorithm 1. In *findfastestpath,* suppose L is the maximum number of iterations of the *while loop*. Then, the value of L is equal to the total number of edges of all the possible paths from the start node to end node. Let $node_e$ be the end node of a graph where Algorithm 1 is applied and let $Edges(node_e)$ be the total number of edges of all possible paths from start node to $node_e$. Hence, $L = Edges(node_e)$. For a given node $node_n$ where $node_n$ is any node in the graph except the start node, $Edges(node_n)$ is given recursively as shown below.

$$\text{Edges}(node_n) = \begin{cases} 1, & c \text{ is a successor of the start node} \\ |S| + \sum_{node_s \, \epsilon S} \text{Edges}(node_s), & \text{otherwise} \end{cases} \quad (30)$$

where S is a set of the successors of $node_n$.

To explain Equation (30), let us consider a simpler graph shown in Figure 9. It is easy to see that the total number of edges of all the possible paths from start node to $node_A$ is 1 since it is a successor of the start node hence, $Edges(node_A) = 1$. Similarly, $Edges(node_B) = 1$ and $Edges(node_C) = 1$. For $node_D$, it can be reached from $node_A$, $node_B$, and $node_C$. Hence, $\text{Edges}(node_D)$ is the summation of $Edges(node_A)$, $Edges(node_B)$, and $Edges(node_C)$ plus

the number of edges from $node_A$ to $node_D$, $node_B$ to $node_D$, and $node_C$ to $node_D$. Therefore, $Edges(node_D) = 6$.

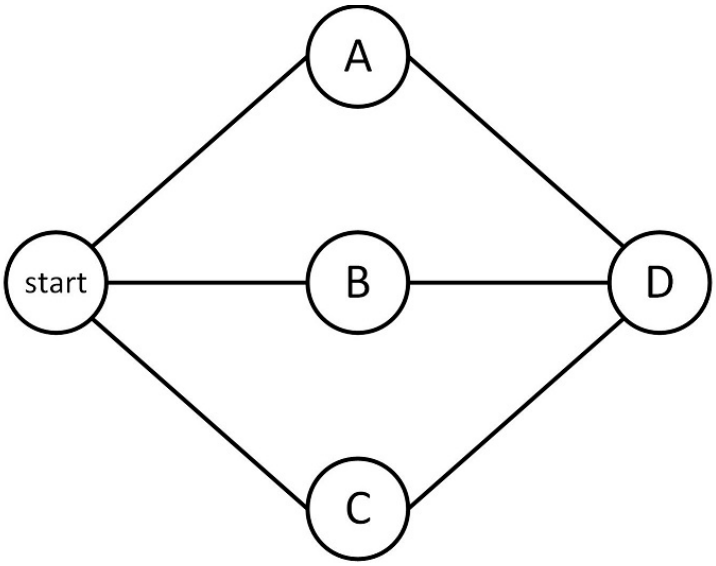


Figure 9. Example graph to illustrate computation of $Edges(node_D)$.

Inside *findfastestpath*, *checkpathtosuccessor* is implemented. In *checkpathtosuccessor*, each cube in *rc* is checked for conflict. Checking for conflict of one cube is $O(log_2 n)$ since it is a standard searching operation in red–black tree where n is the number of UAVs reserved in the cube. Let c^{max} be the maximum number of cubes to be checked between two nodes. Then the complexity of *checkpathtosuccessor* is $O(c^{max} log_2 n)$. Since *checkpathtosuccessor* is called in the *while loop* of *findfastestpath*, the complexity of *findfastestpath* is $O(Lc^{max} log_2 n)$.

As shown in Algorithm 1, *findfastestpath* is called iteratively in a while loop. It is called iteratively until we know that the current path found is the fastest time that the UAV can exit the intersection from the time it arrived at the reservation zone. Suppose I^{max} is the maximum number of iterations of *findfastestpath*. Thus, the complexity of the *while loop* in Algorithm 1 is $O(I^{max} Lc^{max} log_2 n)$.

After finding the path, the cubes will be reserved to the UAV as shown in the last line of Algorithm 1. Reserving a cube is an insertion operation in a red–black tree with complexity of $O(log_2 n)$. Since c^{max} is the maximum number of cubes to be checked between two nodes, the total number of cubes to be reserved in the path from start to end node is $c^{max} e$, where e is the number of edges in the found path. Thus, the complexity of reserving the cubes is $O(c^{max} e log_2 n)$. Therefore, the complexity of Algorithm 1 is $O(I^{max} Lc^{max} log_2 n + c^{max} e log_2 n)$.

Even though L is a constant value, its value is the largest compared to the other variables. In our intersection model, the maximum value of L is 128,101, which is when we find a path for a UAV that will enter the intersection at the second left-most lane, while the maximum value of e is 13. Thus, the overall complexity of scheduling a UAV is $O(I^{max} Lc^{max} log_2 n)$. To make the scheduling faster, we will show how to reduce the complexity of the scheduling algorithm in the next subsection.

7.2. Reducing the Complexity of the Scheduling Algorithm

Using the same scheduling algorithm, we propose another mode to improve the computational complexity of the scheduling algorithm, specifically the complexity of the function *findfastestpath*. In this new mode, we minimize the number of possible paths to be searched by only allowing the UAVs to change layers at the beginning and end of their paths. For example, considering a UAV that will enter the intersection from the left-most lane, same with the example shown in Figure 5, we reduce the number of allowed paths the UAV can take in the new mode as shown in Figure 10. To show the paths clearer, Figure 11 shows Figure 10 in y-z plane. For the remainder of this paper, we will call mode 1 as the

mode where a UAV can change layer anywhere in the intersection and mode 2 as the mode where a UAV can change layer only at the beginning and end of the path in the intersection.

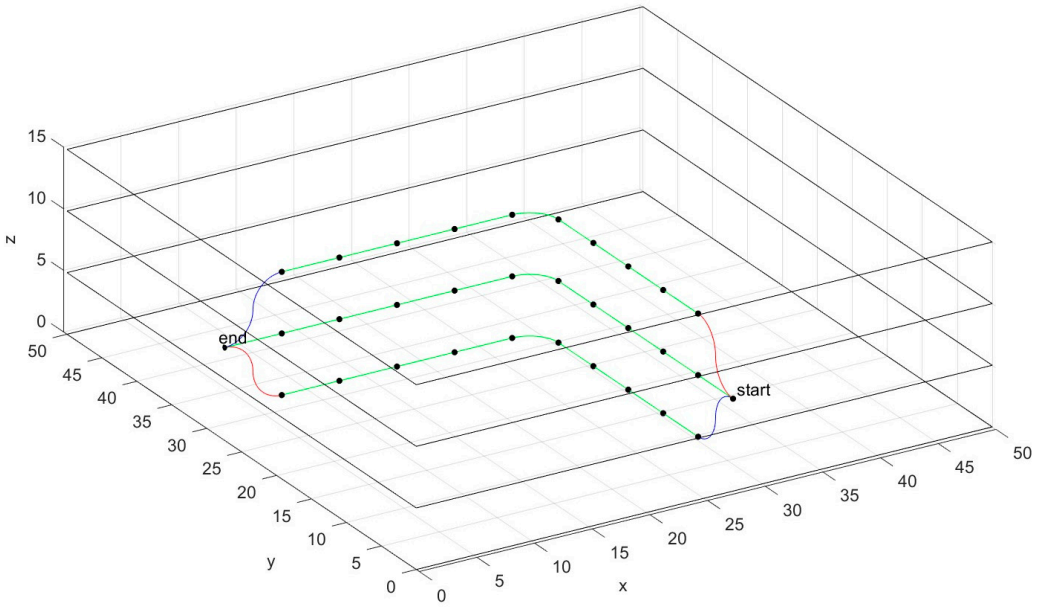


Figure 10. 3D view of the graph to be searched for a UAV entering from the left-most entrance on mode 2 where UAV can change layer only at the start and end of the intersection.

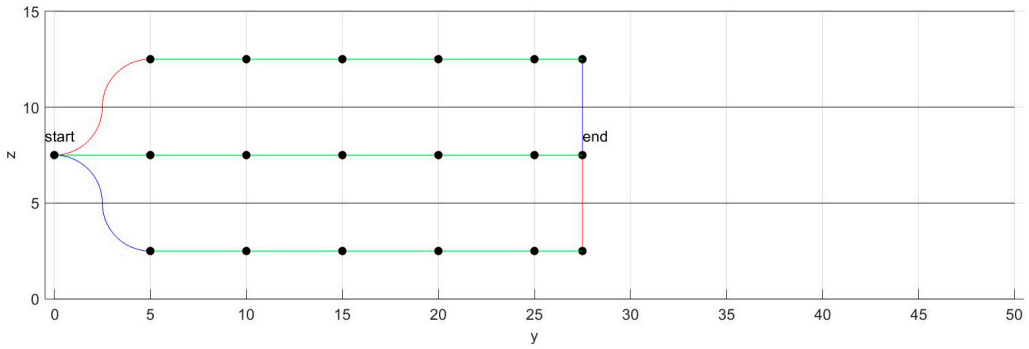


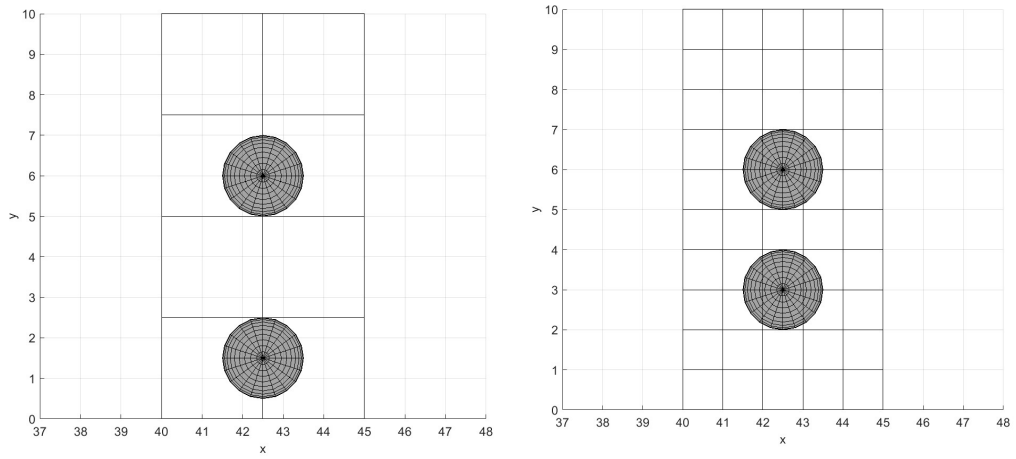
Figure 11. y-z view of Figure 10.

In mode 2, using the same scheduling algorithm *findfastestpath*, a node will only be visited once since there is only one edge connected to each node (except the start and end node). Thus, the worst-case complexity of *findfastestpath* is reduced to $O(Ec^{max}log_2 n)$ where E is the total number of edges in the graph and the maximum value of E is 39, which is much faster compared to the value of L in mode 1 which is 128,101. In Section 8.1, we will show the performance of the system on mode 1 and mode 2.

7.3. Effect of the Cube Size to the Efficiency in the Intersection

As discussed in the complexity analysis of the scheduling algorithm, the complexity of the scheduling increases as the size of the cube decreases. However, there is an advantage to using smaller cube size. For example, in Figure 12a, the size of each cube is 2.5 m and there are two UAVs with radius of 1 m. Since there can be only one UAV in a cube at a time, the minimum distance between them is the size of the cube, which is 2.5 m. On the other hand, if the size of each cube is 1 m, as shown in Figure 12b, then the minimum distance between the UAVs becomes 1 m. Hence, as we decrease the cube size, more UAVs can be in the intersection, thus increasing the efficiency in the intersection. Therefore, there is a

tradeoff between the effect of the cube size to the scheduling run time and the efficiency in the intersection.



(**a**) (**b**)

Figure 12. Top view of the intersection showing the minimum distance between UAVs on different cube sizes. (**a**) Size of a cube is 2.5 m. (**b**) Size of a cube is 1 m.

8. Simulation Results

8.1. Performance of the System in Mode 1 and Mode 2

To compare the performance of the system in mode 1 and mode 2, simulations were run using both modes and the parameters in Tables 2 and 3. The parameters of the intersection model are listed in Table 2, while the scheduling parameters, including the parameters of the Genetic Algorithm are listed in Table 3.

Table 2. Intersection parameters.

Intersection Parameters
s_{min} = 17 m/s
s_{max} = 19 m/s
r_{min} = −3.5 m/s^2
r_{max} = 4 m/s^2
UAV diameter is randomly assigned between 1 to 4 m
l_{rz} = 190 m
l_{qz} = 52 m
l_{az} = 46 m
5 lanes per way
Lane width = 5 m
Intersection layer height = 5 m
Intersection size = 50 m × 50 m
Cube size = 1 m

Table 3. Scheduling parameters.

Scheduling Parameters
Δt = 0.05 s
$k\Delta t$ = 5 s
GA's population size per generation = 100
GA's number of generations = 50
GA's mutation rate = 10%

As listed in Table 2, there are five lanes per way in a four-way intersection. In each mode, simulations were run on different arrival rates per direction and the arrival rates are

identical for all ways. The UAVs are generated with exponentially distributed inter-arrival time. The values of s_{min}, s_{max}, r_{min}, and r_{max} are arbitrarily chosen based on the capabilities of the UAVs nowadays. Since the UAVs in the system can be of different sizes, the sizes of the UAVs in the simulations are randomly assigned between 1 to 4 m. Consequently, since a UAV can be as big as 4 m in diameter, the width of the lane and height of a layer are set to 5 m to allot a minimum space of 0.5 m to each side of the UAVs. The lengths of the reservation zone, queueing zone, and acceleration zone are assigned based on Equations (1), (3) and (4), respectively. A cube size of 1 m is used since the scheduling run time will be longer than $k\Delta t$ if smaller cube size is used. The actual scheduling run time in the simulations will be shown in the future subsection.

For the parameters of the Genetic Algorithm, the GA population size is 100, i.e., a total of 100 chromosomes (sequences of UAVs) are evaluated per generation. The chromosomes are evaluated using the objective function in Equation (21). From the population, the top 50% are selected as the parent chromosomes. Then, crossover is performed to the selected parent chromosomes to create new offspring chromosomes. After generating offspring chromosomes, mutation operation is performed to a chromosome with probability of 10%. Mutation is performed by swapping random UAVs in a chromosome. The algorithm terminates after 50 generations and the top-1 chromosome of the final generation is used as the sequence of UAVs in scheduling. GA is implemented every $k\Delta t$, which is 5 s in the simulation.

The values used for the GA parameters are determined empirically over different simulations. From the simulations, it has been observed that using a larger population size and a larger number of generations gives better results, but also increases the scheduling run time. Thus, the values of these parameters are chosen such that the scheduling run time can be completed before the next scheduling starts. In Sections 8.3.1 and 8.3.2, we provide more results showing the effect of the number of generations on the performance of the system.

The performance of the system in Mode and Mode 2 are compared by the average time spent in the system of the UAVs. The time spent in the system of a UAV is the time spent from a UAV's arrival at the entrance of the reservation zone until it exits the intersection. From the given values of s_{min}, s_{max} , length of the zones in the approaching area, and the intersection size listed in Table 2, the average time in the system of UAVs when they do not experience delay is 20 s. The delay is the additional time a UAV spent in the approaching area due to the queue of UAVs ahead and the time spent waiting at the entrance of the acceleration zone to follow its scheduled time to enter the intersection. For each mode and arrival rate, we ran five simulations and the average of the five simulations are shown in Figure 13.

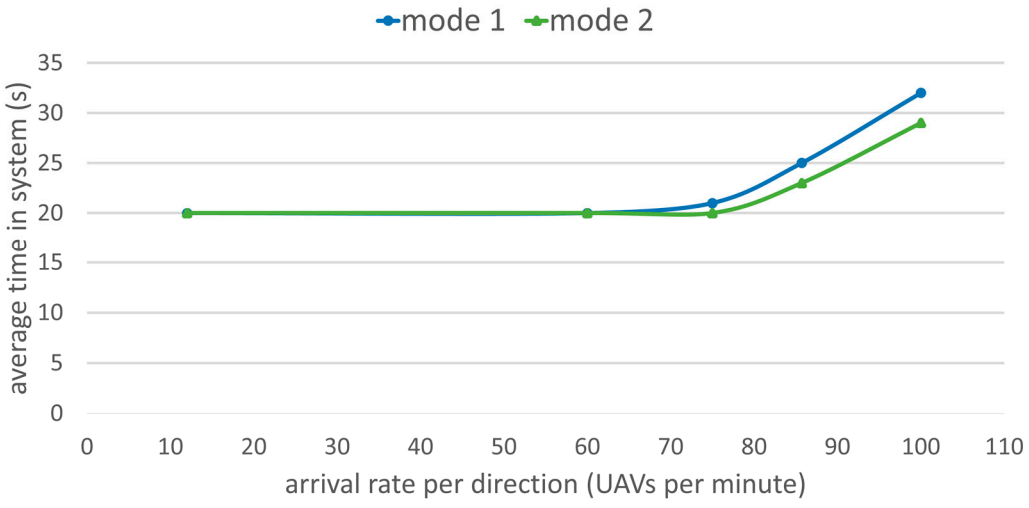


Figure 13. UAVs' average time spent in the system in mode 1 and mode 2.

Figure 13 shows that the average time spent in the system of the UAVs in both modes is 20 s when the arrival rate is at most 60 UAVs per minute. This means there is no delay in the intersection when arrival late is at most 60 UAVs per minute. When the arrival rate is higher than 60 UAVs per minute, the average time spent in the system of the UAVs in mode 2 is always shorter than mode 1.

For the remaining part of this subsection, we illustrate why mode 2 performs better than mode 1. Consider scheduling 20 UAVs at the scheduling instant t_i where one UAV is located at each lane and there are no other reservations in the intersection. The resulting trajectories of the UAVs using mode 1 are shown in Figure 14a, while the resulting trajectories using mode 2 are shown in Figure 14b. The spheres in the figure represent the center position of the UAVs. One UAV is represented by one color and UAVs coming from the same way have the same color. Note that the time is not reflected in the figure hence there are no collisions, even though there are more than one UAV at the same space in the figure.

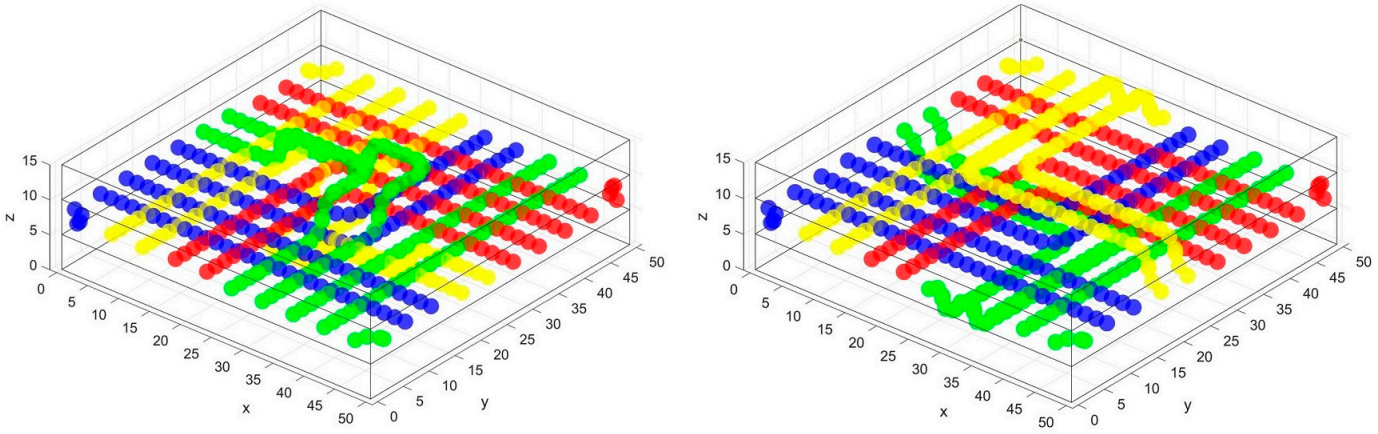


Figure 14. Trajectories of the UAVs scheduled at t_i in (**a**) mode 1 and (**b**) mode 2. One UAV is represented by one color and UAVs coming from the same way have the same color.

Since there are no other reservations in the intersection at t_i, all the UAVs will not experience delay in mode 1 and mode 2. Also, the same UAVs will change layer once both in mode 1 and mode 2. Hence, the total time in the system of the UAVs is the same in both modes. The trajectories of the UAVs that will change layer are different between mode 1 and mode 2. Figure 15a shows the trajectories of the UAVs that will change layer in Figures 14a and 15b shows the trajectories of the UAVs that will change layer in Figure 14b. As expected in mode 2, shown in Figure 15b, the UAVs only changed layers at the start and end of the intersection.

At the next scheduling instant t_{i+1}, suppose there are new 20 UAVs to be scheduled and there is one UAV coming from every entrance lane. The UAVs scheduled at t_i are still in the intersection at the time the UAVs to be scheduled at t_{i+1} will enter the intersection. Thus, the trajectories of the UAVs scheduled at t_i will affect the availability of the cubes for the UAVs to be scheduled at t_{i+1}.

The trajectories of all the UAVs scheduled at t_{i+1} are shown in Figure 16a,b for mode 1 and mode 2, respectively. As a result of the scheduling at t_{i+1}, ten UAVs will change layer in mode 1 while six UAVs will change layer in mode 2 to avoid collision with other UAVs. To show the trajectories of the UAVs that will change layer clearer, the trajectories of UAVs that will change layer at t_{i+1} in mode 1 are shown in Figure 17a, while the trajectories of the UAVs that will change layer at t_{i+1} in mode 2 are shown in Figure 17b.

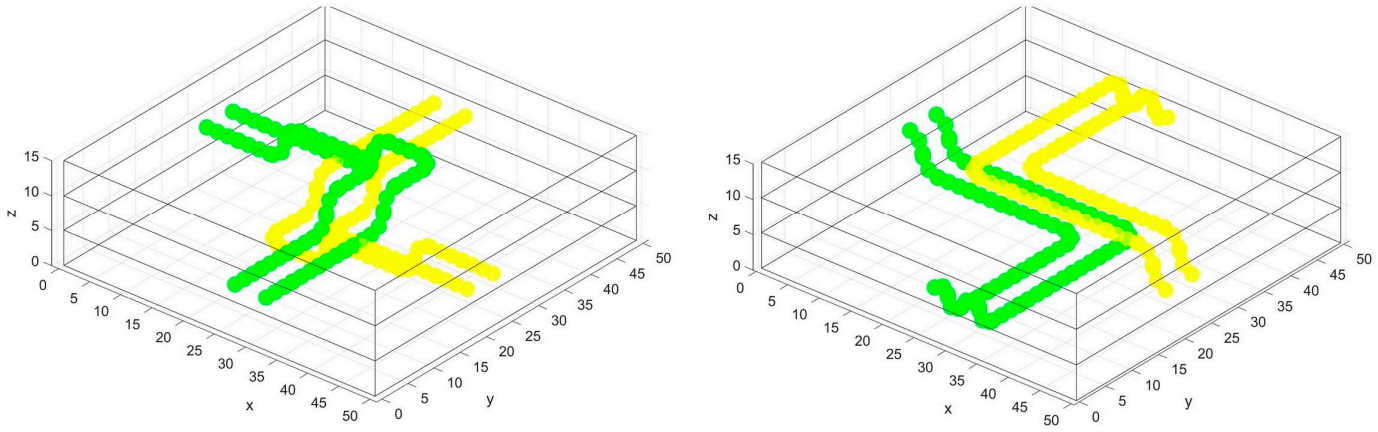


Figure 15. Trajectories of the UAVs scheduled at t_i that will change layer in (**a**) mode 1 and (**b**) mode 2. One UAV is represented by one color and UAVs coming from the same way have the same color.

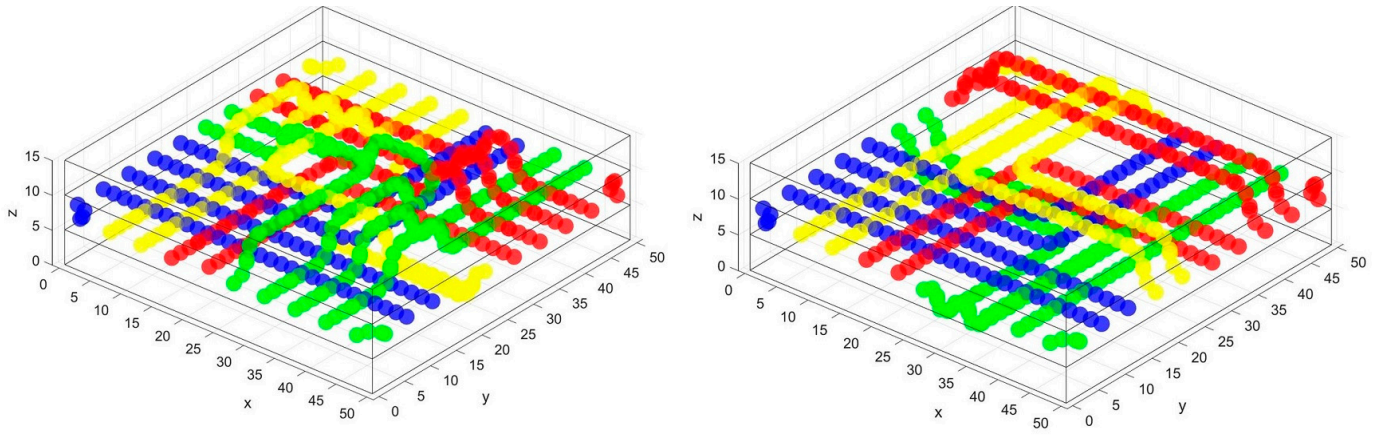


Figure 16. Trajectories of the UAVs scheduled at t_{i+1} in (**a**) mode 1 and (**b**) mode 2. One UAV is represented by one color and UAVs coming from the same way have the same color.

From Figure 17, we can see that lesser number of UAVs will change layer in mode 2 at t_{i+1}. This is because the trajectories of the UAVs that will change layer at t_i, as shown in Figure 15b, allow more UAVs scheduled at t_{i+1} to travel in the intersection without needing to change layers.

The travel time in the intersection of a UAV increases as the number of times it changes layer increases. Consequently, the total time in the system of the UAVs increases as the number of UAVs that change layers increases. Therefore, the total time spent in the intersection of the UAVs scheduled at t_{i+1} using mode 1 is longer than mode 2 since more UAVs will change layer in mode 1.

Since the objective of the scheduling is to minimize the total time spent in the system of the UAVs being scheduled, UAVs can change layer anywhere in the intersection in mode 1 as long as the total travel time is minimized. However, it can be observed that it is better for a UAV to change layer at the start and end of its path in the intersection because this gives higher chance for other UAVs that will be scheduled at a later scheduling time to travel in the intersection without changing layer, thereby, decreasing the total travel time of the UAVs in the next scheduling time. Therefore, instead of only minimizing the total travel time of the UAVs being scheduled, minimizing the possible conflicts of their trajectories

with UAVs that will be scheduled at a later scheduling time must also be considered. In conclusion, it is better to schedule the UAVs using mode 2 compared to mode 1 since mode 2 improves the complexity of the scheduling and utilizes the intersection more efficiently.

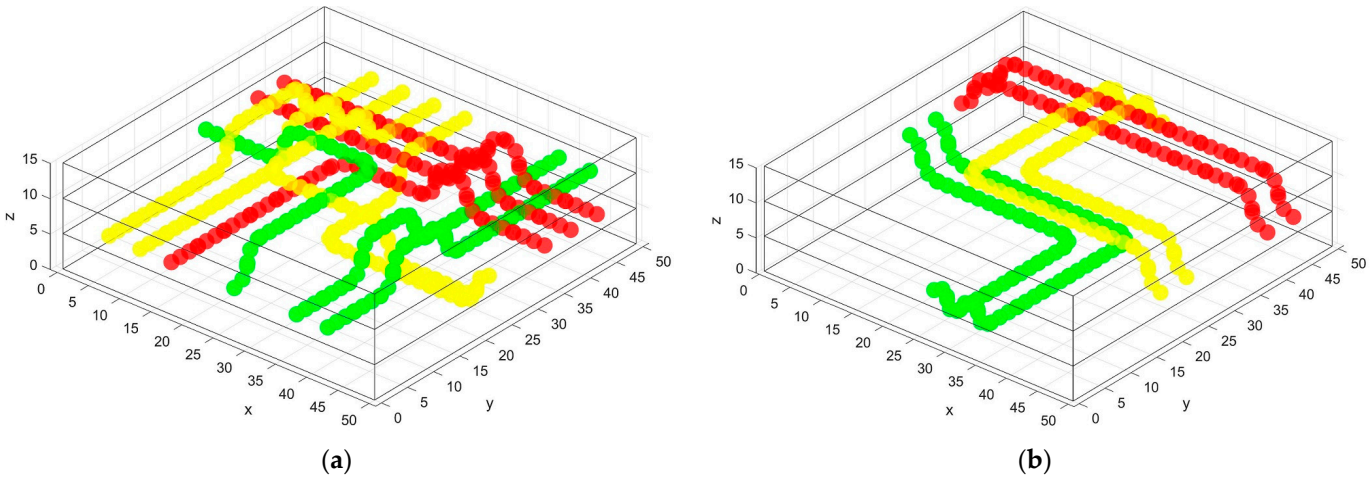


Figure 17. Trajectories of the UAVs scheduled at t_{i+1} that will change layer in (**a**) mode 1 and (**b**) mode 2. One UAV is represented by one color and UAVs coming from the same way have the same color.

8.2. Sample Trajectories of UAVs

From the simulations, all UAVs were able to find paths and travel across the intersection without colliding with other UAVs. To show the trajectories of the UAVs, a simulation was run using mode 2, the parameters in Tables 2 and 3, and the arrival rate of 100 UAVs per minute per direction. Figure 18 shows the combined trajectories of UAVs entering the intersection within an arbitrarily chosen 5 s time window in the simulation. Similar to Figures 14–17, the size of the spheres in Figure 18 is smaller than the actual size of the UAVs for clearer illustration of the trajectories. One UAV is represented by one color and UAVs coming from the same way have the same color. The time is not reflected in the figure, hence there is no collision between the UAVs, even though there are colliding spheres in the figure.

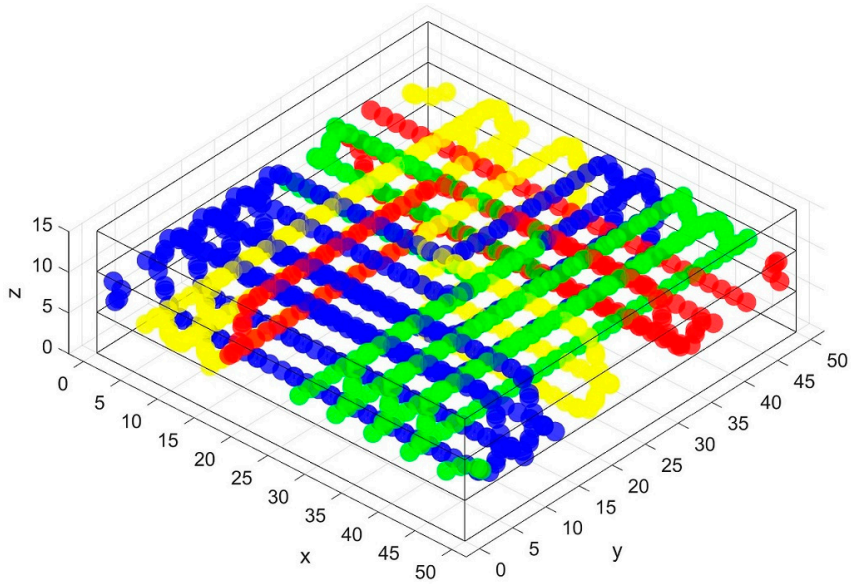


Figure 18. Sample trajectories of the selected UAVs in the simulation. One UAV is represented by one color and UAVs coming from the same way have the same color.

8.3. Evaluation of the Scheduling Sequence

8.3.1. Comparison of our Proposed System with FCFS Scheduling

To show the performance of our proposed system to the total travel time of the UAVs in the intersection, we compared the performance of our scheduling scheme with First Come First Serve (FCFS) scheduling. In FCFS scheduling, the UAVs are scheduled in the order their requests are received by the intersection manager. This is different with our scheduling that uses genetic algorithm where the UAVs are scheduled based on the best sequence obtained using the objective function. We compared their performances through the average time in the system of the UAVs. We used mode 2 and the parameters in Tables 2 and 3 except for the number of generations, G, used in genetic algorithm. We ran simulations on two values of G, which are 50 and 80. As previously mentioned, the average time in the system of UAVs when they do not experience delay is 20 s when the parameters in Table 2 are used.

The results of the simulations are shown in Figure 19. We can see that our scheduling scheme that uses genetic algorithm (GA) always performs better than FCFS. It also shows that the average time in system decreases when bigger value of G is used. However, there is no significant improvement in the performance when the value of G is higher than 80 using the simulation parameters in Tables 2 and 3, thus we only showed results when values of G are 50 and 80. Compared to FCFS, our scheduling scheme decreases the average time in the system by 27% when G = 80 and the arrival rate is 100 UAVs per direction.

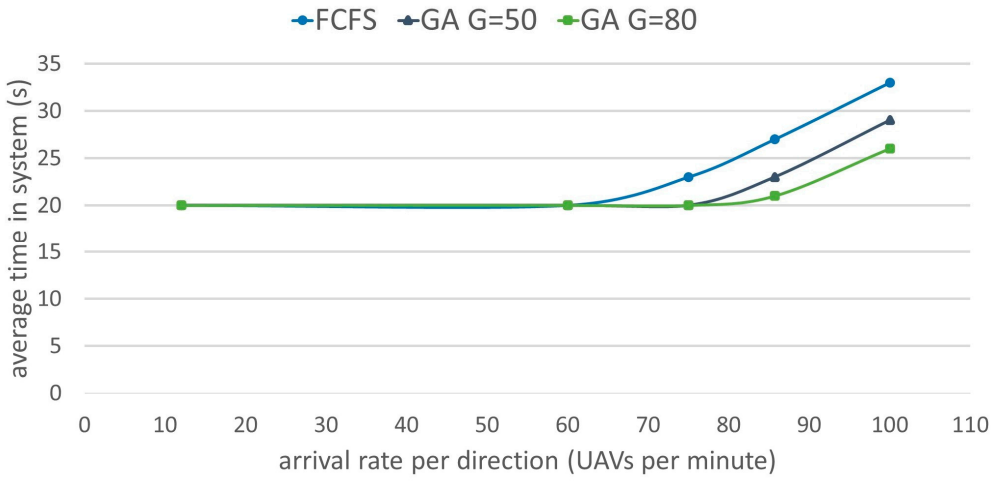


Figure 19. UAVs' average time in the system using our proposed system and using FCFS scheduling.

As shown in Figure 19, the increase in the average time in the system is higher in FCFS when arrival rate is higher than 70 UAVs per minute. A problem with FCFS is the intersection is not efficiently utilized when the traffic is heavy since the UAVs from different ways cross the intersection in the order of their arrivals. This causes more interruptions to the flow of traffic in the intersection and thus increases the average time spent of the UAVs in the system and decreases the throughput in the intersection. On the other hand, our scheduling scheme that uses genetic algorithm processes multiple UAVs in scheduling and it allows the UAVs to enter the intersection by group resulting to less interruptions to the paths of the UAVs.

8.3.2. Run Time of the Scheduling Algorithm

The run time of our scheduling algorithm using mode 2 and the parameters in Tables 2 and 3 on a CPU with Intel Core i5-10210U at 1.60 GHz clock speed is shown in Figure 20. As discussed in the complexity analysis of our scheduling algorithm, the complexity of the algorithm increases as the cube size decreases and as the arrival rate increases. When the arrival rate is high, i.e., 100 UAVs per minute per direction, the run time of the scheduling

algorithm is 2 s when $G = 50$ and 3.53 s when $G = 80$. Since $k\Delta t$ is set to 5 s, there is enough time to schedule the UAVs even when arrival rate is high and $G = 80$.

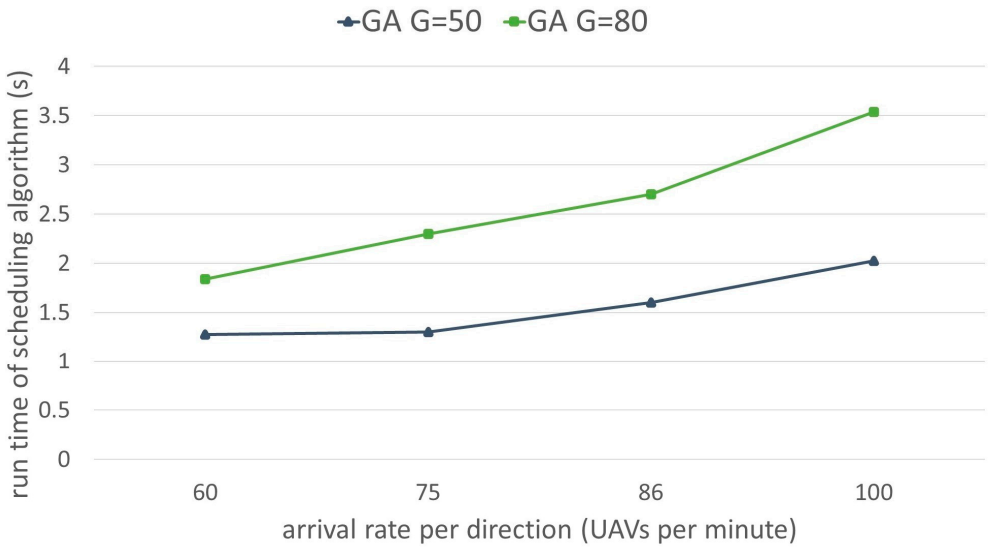


Figure 20. Average scheduling run time of UAVs every $k\Delta t$ using mode 2.

8.3.3. Maximum Time Spent per Lane of the UAVs

To better understand and interpret the results of the scheduling, we provide more results showing the maximum time spent of the UAVs per lane and the number of UAVs that crossed the intersection per lane.

The maximum time in system per lane when $G = 80$ and using mode 2 and parameters in Tables 2 and 3 is shown in Figure 21. Let lane 1 be the left-most lane, let lane 2 be the second left-most lane, let lane 3 be the middle lane, let lane 4 be the second right-most lane, and let lane 5 be the right-most lane. Referring to Figure 3, UAVs that will turn left in the intersection can take either lane 1 or lane 2 while UAVs that will move straight in the intersection can take either lane 3 or lane 4, and UAVs that will turn right must take lane 5.

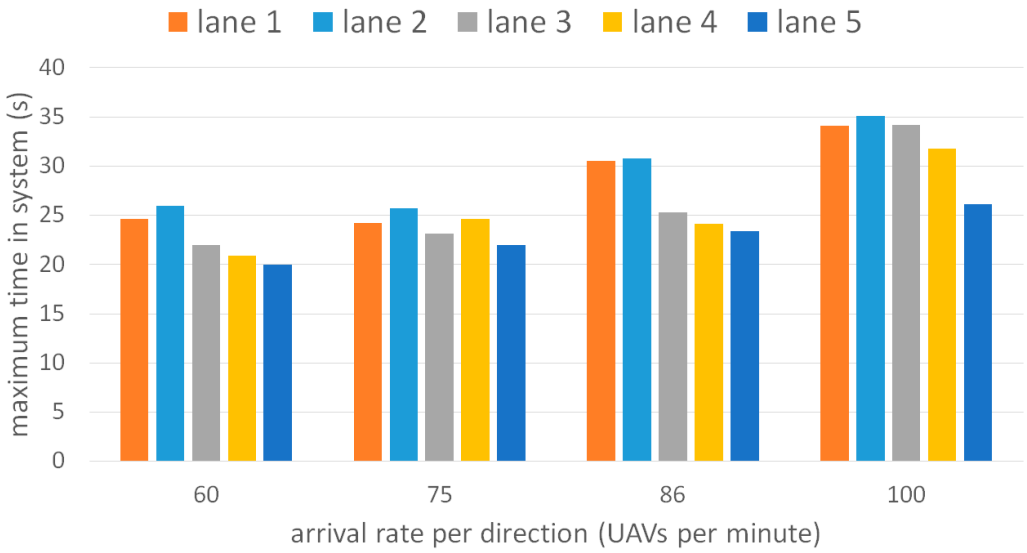


Figure 21. Maximum time in system of UAVs per lane using mode 2 and $G = 80$.

As shown in Figure 21, UAVs in lane 1 and lane 2 mostly experience longer maximum time in the system compared to the UAVs in other lanes. Aside from the reason that the trajectories of the UAVs in lane 1 and lane 2 are longer since they are left-turning UAVs,

they also have higher possibility of conflict in the intersection compared to the straight moving and right turning UAVs, resulting in longer waiting time before they can enter the intersection. Consequently, the UAVs in lane 1 and lane 2 mainly contribute to the increase in the average time in the system as the arrival rate increases.

8.3.4. Number of UAVs That Crossed the Intersection per Lane

From the same simulations in Figure 21, we also obtained the number of UAVs that crossed the intersection per lane within a randomly chosen 1-min time window in the simulation. As shown in Figure 22, we can see that highest number of UAVs entered from lane 5. This is because there is no conflict with UAVs from other ways for right turning UAVs. Hence, UAVs in lane 5 can enter the intersection as soon as there is no conflict with the UAV ahead. On the other hand, UAVs in lane 1 to 4 should avoid possible collisions with UAVs entering from the other ways. Moreover, we can also see that UAVs from lanes 1 to 4 were also able to cross the intersection without significant differences in the number of UAVs, which indicates that there is no starvation experienced by UAVs from any of the lanes.

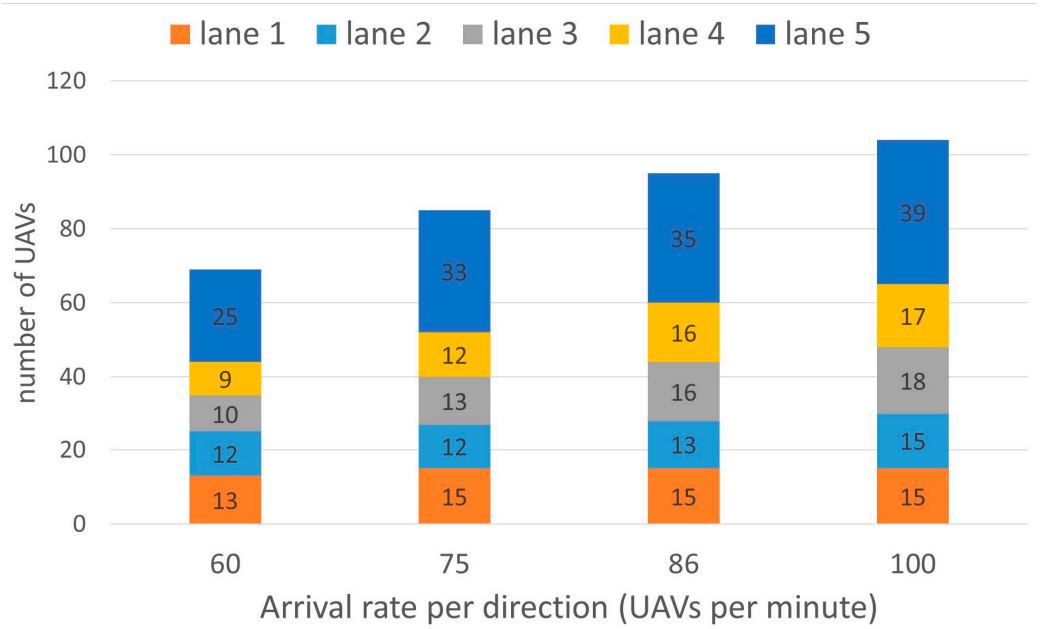


Figure 22. Number of UAVs that crossed the intersection per lane within 1 min using mode 2 and $G = 80$.

8.4. Comparison of Our Proposed System with the 2D Optimal Static Traffic Light Intersection System

We also compared the performance of our system with an optimal static traffic light system in 2D roads. In the statically optimal traffic light system, the time durations of the green, yellow, and red lights of a traffic signal are fixed and optimized based on the average arrival rates of the vehicles. Their performances are compared using the queueing delay and the average number of UAVs in the intersection. Queueing delay is measured as the additional time spent in the system due to queue of UAVs. In addition, we evaluated their performances on two scenarios. In Scenario 1, all UAVs move straight in the intersection, while in Scenario 2 a UAV can turn left, move straight, or turn right in the intersection with equal probabilities.

To show a fair comparison with the 2D optimal static traffic light system, we applied our scheduling scheme to a 2D intersection model. In the 2D intersection model, there are three lanes per way. Figure 23a shows the intersection model for Scenario 1 and Figure 23b shows the intersection model for Scenario 2. In Scenario 1, since all UAVs move straight,

a UAV can travel on any lane. The blue arrows represent the paths of the UAVs from the north and south ways, while the black arrows represent the paths from the east and west ways. In Scenario 2, left turning UAVs can only travel along the left-most lane, straight moving UAVs can travel on the middle lane or right-most lane, and right turning UAVs can travel on the right-most lane. This is applied to all ways.

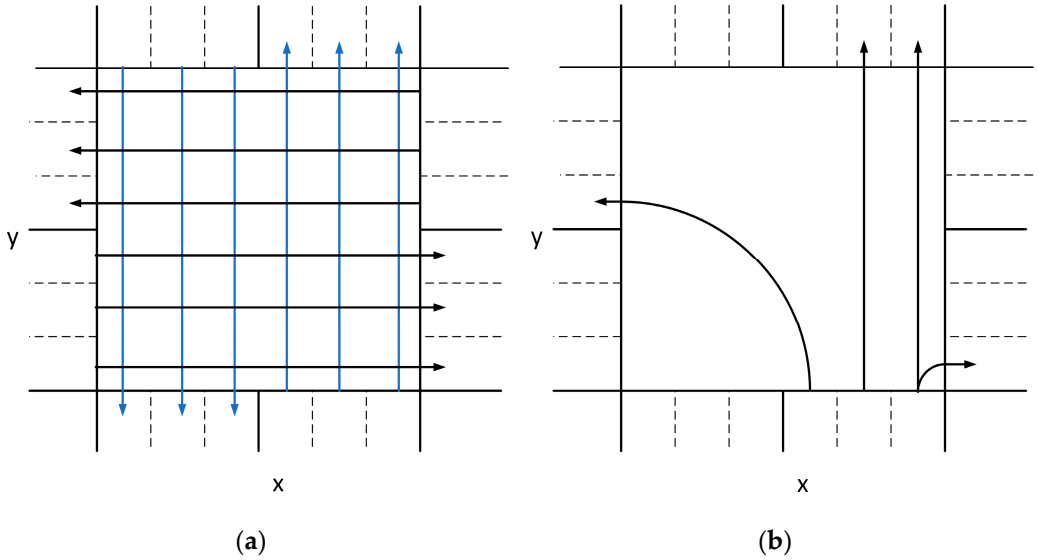
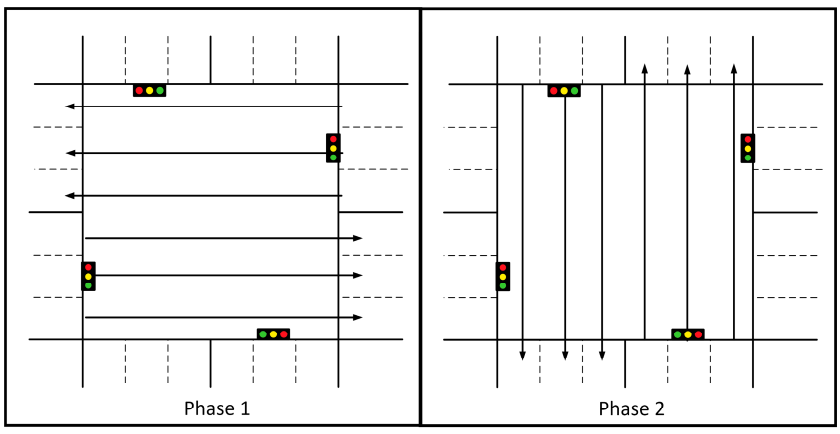


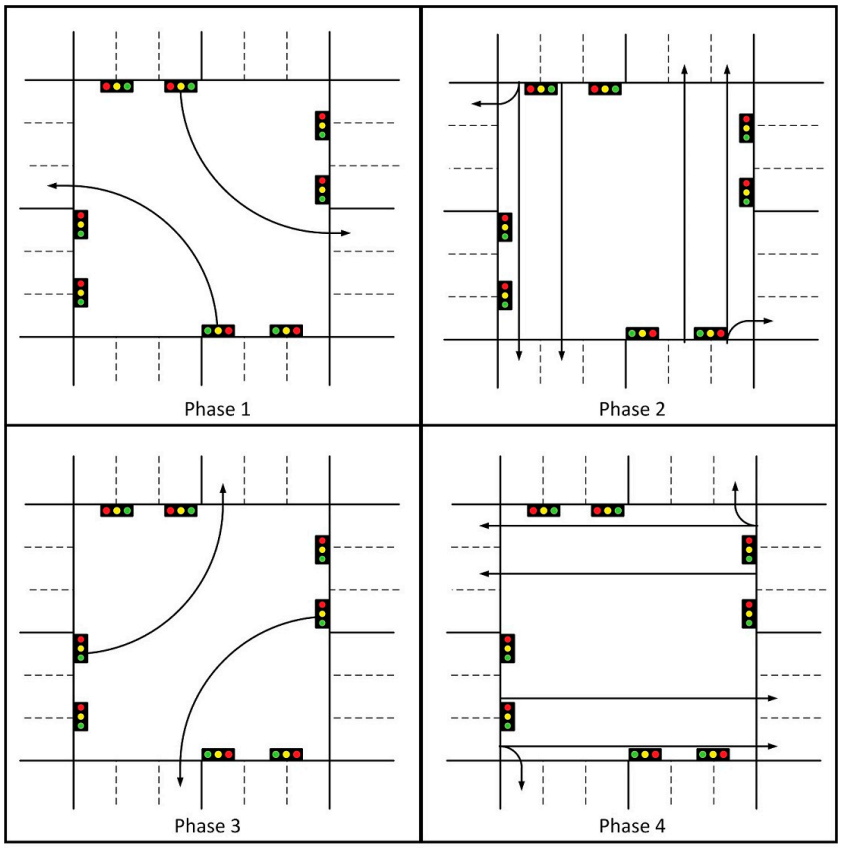
Figure 23. 2D intersection model of (**a**) Scenario 1: when all UAVs move straight and (**b**) Scenario 2: when UAVs can turn left, move straight, or turn right. The blue arrows represent the paths of the UAVs from the north and south ways, while the black arrows represent the paths from the east and west ways.

In the statically optimal traffic light system, a phase indicates the traffic signals that have the same cycle start time and cycle length. In Scenario 1, since all UAVs move straight, there is one traffic signal for each way and there are two phases as shown in Figure 24a. In phase 1 of Figure 24a, the UAVs from the west and east ways are allowed to enter the intersection while in phase 2 of Figure 24a, the UAVs from the north and south ways are allowed to enter the intersection. In Scenario 2, there are two traffic signals for each way because there are left turning UAVs. For each way, one traffic signal is used for the left-most lane while another traffic signal is used for both the middle and right-most lane. Figure 24b shows the four phases of Scenario 2.



(**a**)

Figure 24. *Cont.*



(**b**)

Figure 24. Traffic phases of the traffic light system of (**a**) Scenario 1: when all UAVs move straight and (**b**) Scenario 2: when UAVs can turn left, move straight, or turn right.

The traffic light system simulations are implemented using Table 4 and our proposed system is implemented using the simulation parameters in Tables 4 and 5. In our system, the intersection is divided into 2D sections which we will denote as cells instead of cubes. Also, the UAVs can only travel at a constant speed in the intersection to be fair with the traffic light system.

Table 4. Intersection parameters used in traffic light system and in our proposed system.

Intersection Parameters
$s_{min} = 15$ m/s
$s_{max} = 15$ m/s
UAV diameter = 3 m
Lane width = 5 m
3 lanes per way
Intersection size = 30 m $\times$ 30 m

Scenario 1: All UAVs move straight in the intersection

The queueing delays on this scenario are shown in Figure 25. In the traffic light system, there is a delay even when traffic is low because UAVs cannot enter the intersection, even when there are no UAVs traveling in the intersection from the other ways, when the traffic light on their lanes are yellow or red. For example, when the arrival rate is low, that is six UAVs per minute, every traffic light has a green time duration of 4 s and a yellow time duration of 5 s. Thus, when a UAV is the first in the queue to enter the intersection, then it can wait up to 9 s before it enters the intersection. While when a UAV is in queue and there

are UAVs ahead, its queueing delay is the delay of the UAV ahead plus the time it will take to reach the entrance of the intersection.

Table 5. Parameters used in our proposed system.

Intersection Parameters
$ar_{min} = -3.5$ m/s^2
$ar_{max} = 4$ m/s^2
$l_{rz} = 150$ m
$l_{qz} = 33$ m
$l_{az} = 29$ m
Cell size = 1 m
$\Delta t = 0.05$ s
$k\Delta t = 5$ s
GA population size per generation = 100
GA termination = 80 generations
GA mutation rate = 10%

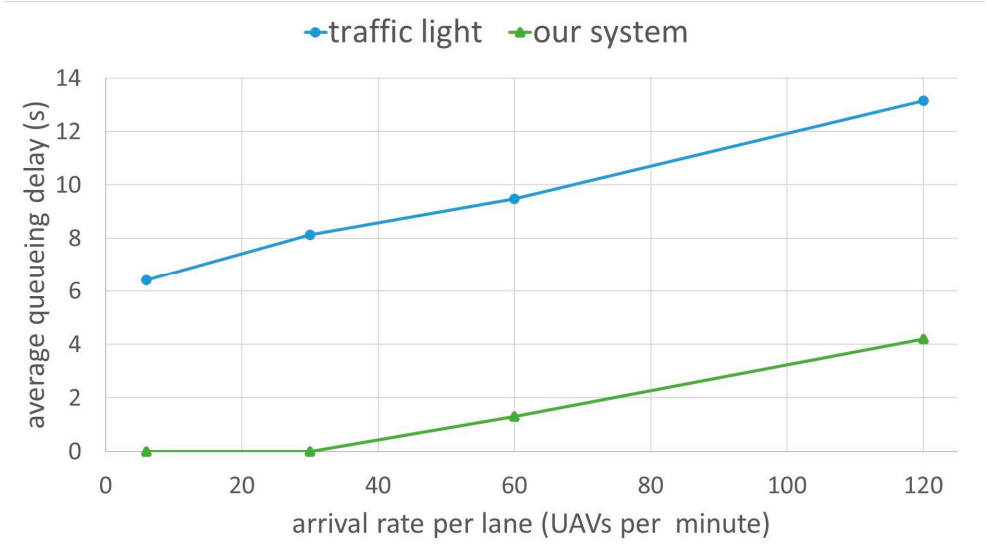


Figure 25. Average queueing delay of the UAVs using 2D optimal static traffic light and using our proposed system in 2D intersection on Scenario 1.

In our scheduling scheme, when traffic is low, UAVs can enter the intersection without stopping if there is no conflict with the paths of the UAVs in the intersection. As shown in the figure, even when traffic is high, our system performs better than traffic light system on this scenario.

The average number of UAVs in the intersection on this scenario are shown in Figure 26. As shown in the figure, the number of UAVs in the intersection using our system is always greater than the traffic light. The difference in the number of UAVs in the intersection is between one to three UAVs. The maximum of average number of UAVs in the intersection is 33 UAVs for the traffic light and 36 UAVs for our system.

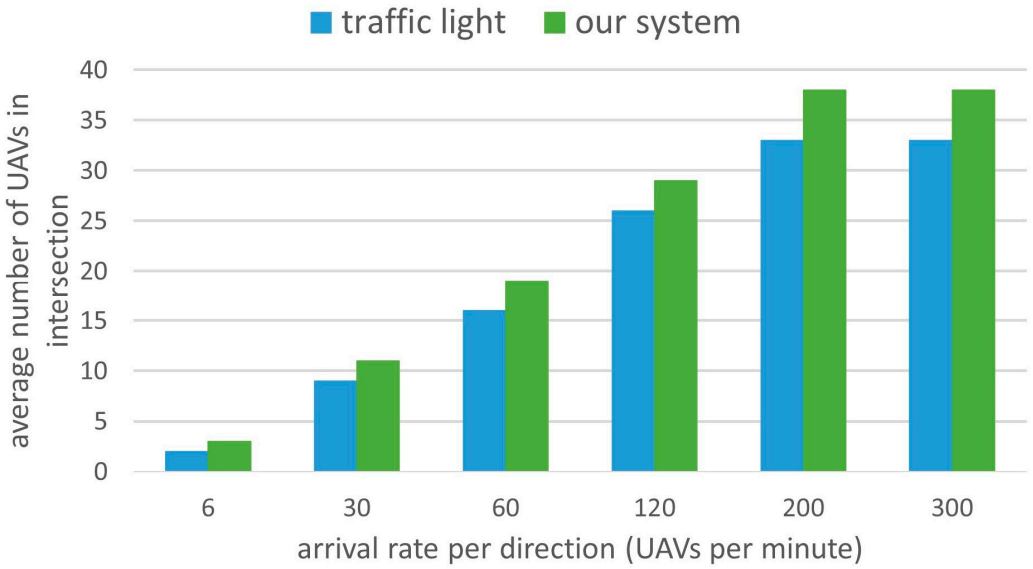


Figure 26. Average number of UAVs in the intersection using 2D optimal static traffic light and using our proposed system in 2D intersection on Scenario 1.

To understand better how both system works, we obtained the number of UAVs in the intersection every 1 s for the traffic light system and every 0.2 s in our system. The results are shown in Figure 27. We ran five simulations for each system. The arrival rate is 200 UAVs per minute per direction in each simulation. We started obtaining the number of UAVs for 25 s after the simulation has been running for 2 min, which is when the simulation is already in a steady state. We can see from the figure that in the traffic light system, there are at maximum 60 UAVs in the intersection. The number of UAVs mostly remained at 60 UAVs. Then, it decreased when the traffic light turned yellow, and the number of UAVs started to increase again to 60 UAVs after the light of the next phase turned green. In our system, the number of UAVs in the intersection changes between 19 UAVs and 60 UAVs. This graph reflects the average number of UAVs when arrival rate is 200 UAVs, as shown in Figure 26.

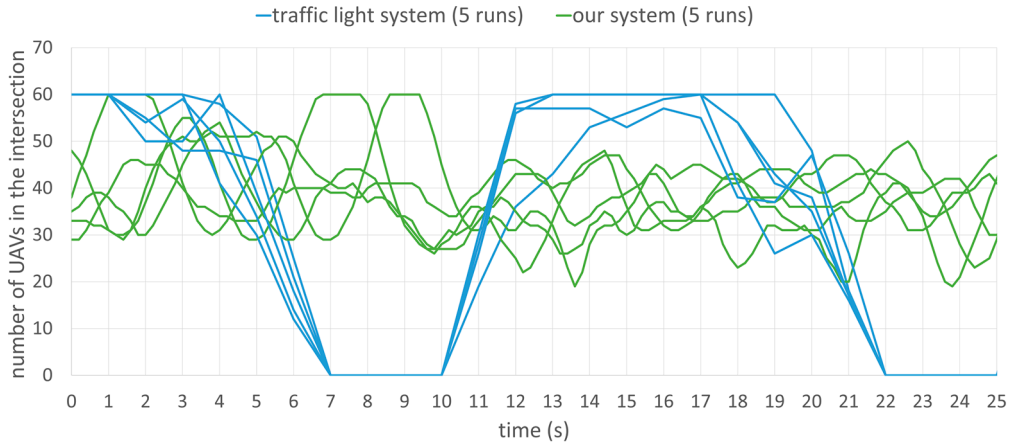


Figure 27. Number of UAVs in the intersection for Scenario 1 when arrival rate is 200 UAVs per minute.

Scenario 2: UAVs can turn left, move straight, or turn right in the intersection

The queueing delays on this scenario are shown in Figure 28. Similar with the previous scenario, UAVs experience delay in the traffic light system even when the traffic is low, since UAVs must wait for the traffic light to be green before they can enter the intersection. In this scenario, when the arrival rate is six UAVs per minute, every traffic light has a green

time duration of 5 s and a yellow time duration of 5 s. Since there are four phases in this scenario, a UAV at the start of a queue can wait up to 30 s before it can enter the intersection on this arrival rate. As shown in Figure 28, our system performs better than traffic light system on low and medium traffic for the same reason with Scenario 1. However, when the arrival rate is greater than 40 UAVs per direction per minute, the delay of the UAVs in the left most lanes contribute to the increase in the average delay of the UAVs. The UAVs turning left experience greater delay than the straight moving UAVs since the trajectories of left turning UAVs experience more conflicts in the intersection. Similar trends will be observed when the arrival rate is higher than 50 UAVs per minute. Meanwhile, in the traffic light system, the duration of the green time is fixed, hence the delays experienced by all lanes are equally distributed.

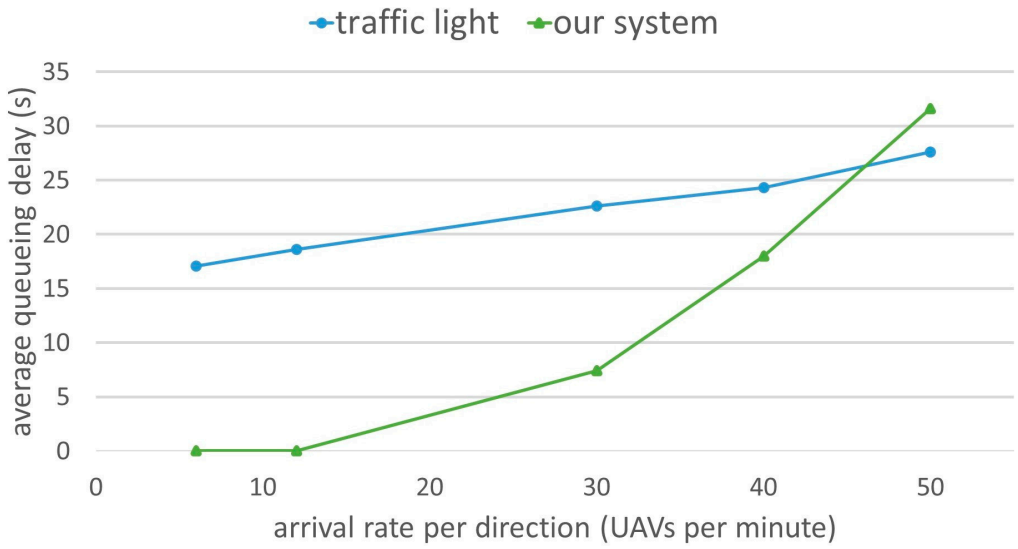


Figure 28. Average queueing delay of the UAVs using 2D optimal static traffic light and using our proposed system in 2D intersection on Scenario 2.

The comparison of the average number of UAVs in the intersection on this scenario is shown in Figure 29. There is no difference using our system and the optimal traffic light system when the arrival rate is less than or equal to 40 UAVs per minute. While when the arrival rate is 50 UAVs per minute, the average number of UAVs in the intersection using the traffic light is greater by one UAV compared to our system. The maximum of average number of UAVs in the intersection is 16 UAVs for the traffic light and 14 UAVs for our system.

Similar to the previous scenario, we also obtained the number of UAVs in the intersection every 1 s for the traffic light system and every 0.2 s in our system, as shown in Figure 30. The arrival rate in each simulation is 50 UAVs per minute per direction. We started obtaining the number of UAVs for 25 s after the simulation has been running for 2 min. Looking at the traffic light system, there are mostly 60 UAVs for the first 3 s, then it decreased after the traffic light turned yellow. The number of UAVs started to increase again when the light of the next phase turned green. However, the maximum number of UAVs during the next phase only reached 20. This is because the green light is assigned to phase 3, as shown in Figure 24b. On phase 3, the UAVs entering the intersection are from the left-most lanes. In our system, the maximum number of UAVs is 29, while the minimum is four UAVs. This also reflects the average number of UAVs when arrival rate is 50 UAVs, as shown in Figure 29.

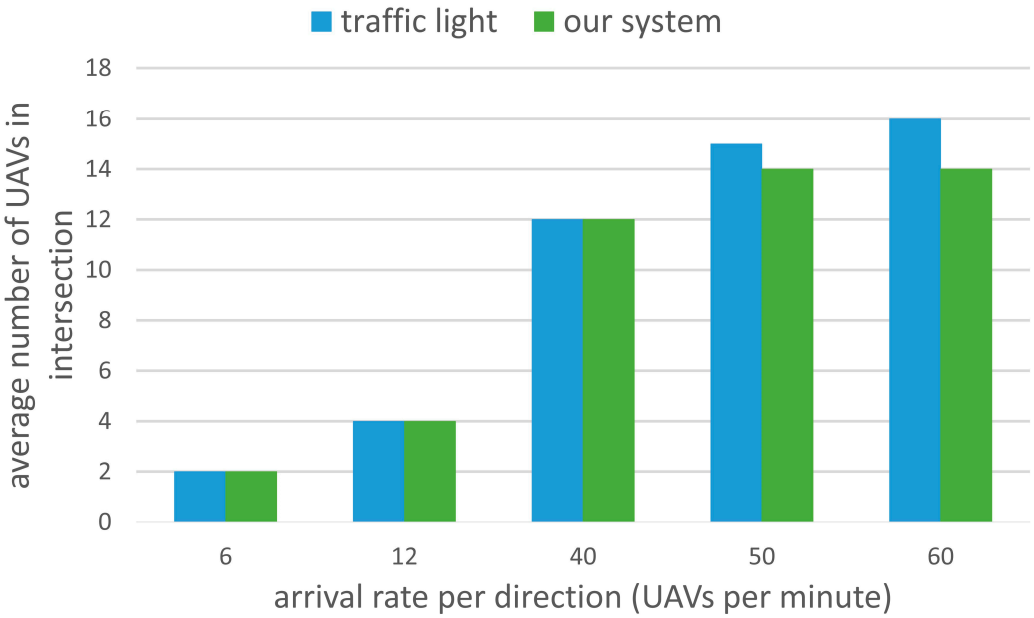


Figure 29. Average number of UAVs in the intersection using 2D optimal static traffic light and using our proposed system in 2D intersection on Scenario 2.

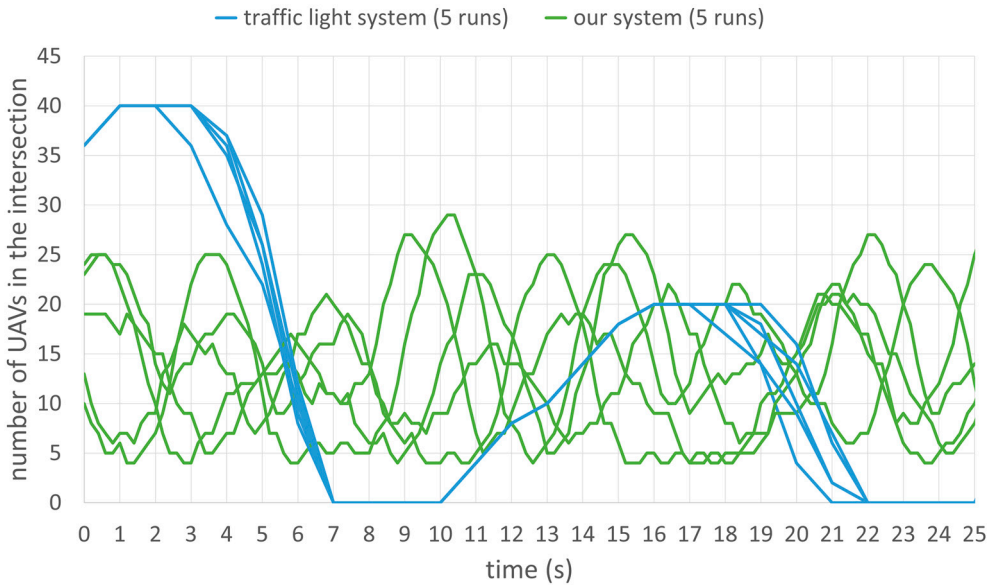


Figure 30. Number of UAVs in the intersection for Scenario 2 when arrival rate is 50 UAVs per minute.

8.5. Efficiency of Our System on Different Values of the Allowed Speed Range in the Intersection

We obtain the efficiency in the intersection on different values of the allowed speed range by getting the maximum number of UAVs in the intersection at a time for the whole simulation time when arrival rate is high, that is 110 UAVs per direction per minute. We used mode 2 and the simulation parameters in Tables 2 and 3, except for the values of s_{min} and $G = 80$. We used fixed value of $s_{max} = 19$ m/s and different values of s_{min} to show the effect of the speed range $[s_{min}, s_{max}]$ to the efficiency. The results are shown in Figure 31. It can be seen from the figure that the efficiency decreases as the speed range increases. The maximum number of UAVs in the intersection is highest at 72 UAVs when the value of s_{min} is equal to s_{max}. Decreasing the value of s_{min} increases the reservation time duration of the UAVs in the cubes resulting to longer distances between UAVs and less UAVs in the

intersection. Even though a UAV moves at s_{max}, the cubes are reserved such that the UAV can occupy the cubes when it travels at any speed from s_{min} to s_{max}.

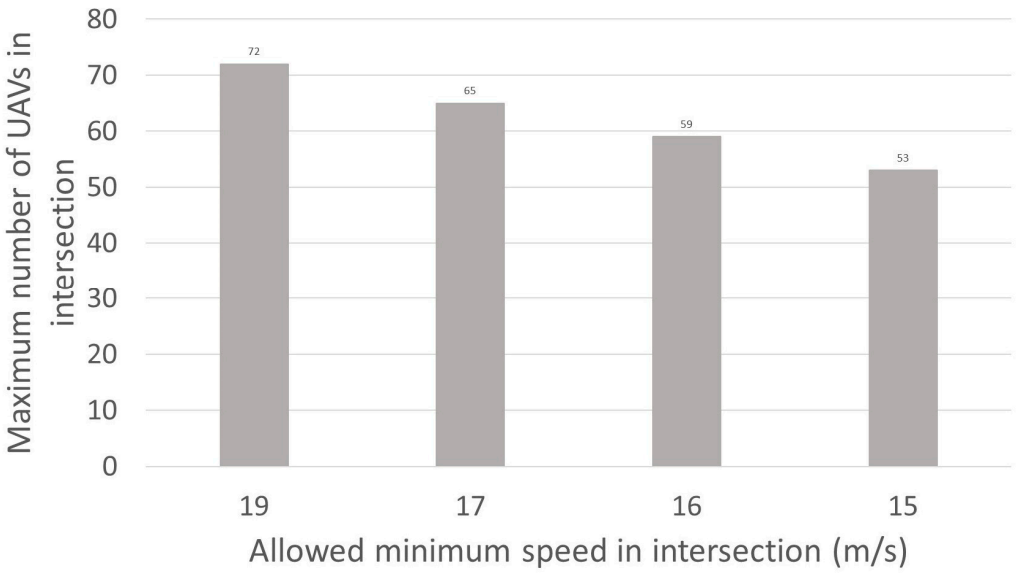


Figure 31. Maximum number of UAVs in the intersection when $s_{max} = 19$ m/s for different values of s_{min}.

In addition, to visualize how congested the intersection is when there are 72 UAVs in the intersection, which happened when s_{min} and s_{max} are both 19 m/s, a snapshot of the intersection showing the positions of the 72 UAVs is shown in Figure 32.

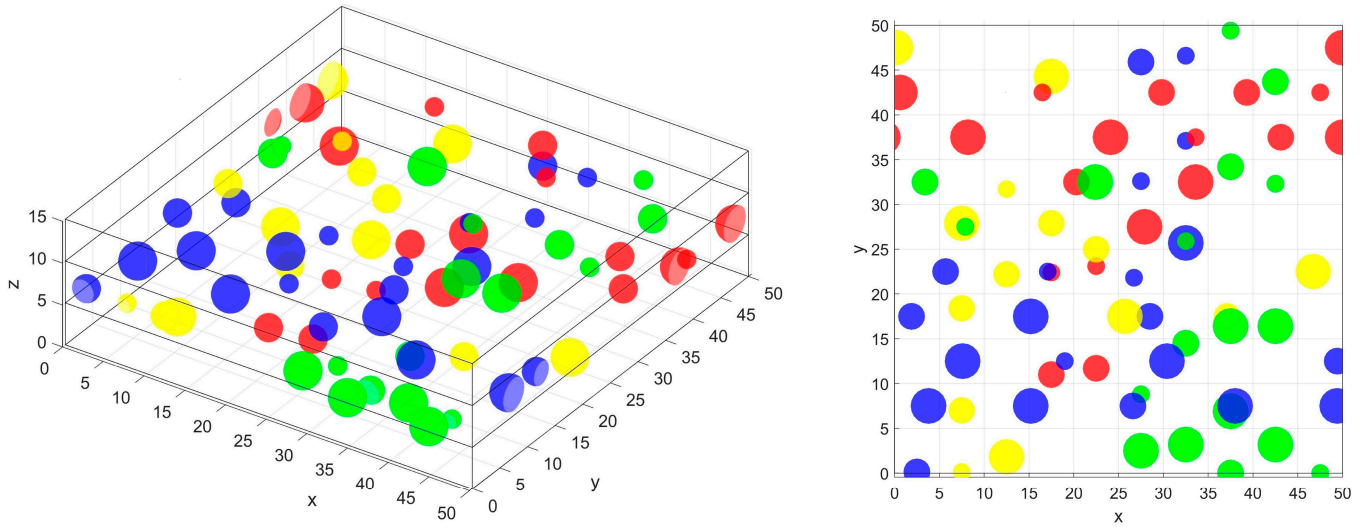


Figure 32. Snapshot of the intersection when there are 72 UAVs in the intersection in (**a**) 3D view and in (**b**) x-y view. One UAV is represented by one color and UAVs coming from the same way have the same color.

9. Conclusions

We proposed a 3D intersection model for UAV traffic in the sky above an urban area. Our intersection model consists of a 3D intersection and approaching areas. We were able to define the behavior of the UAVs in the approaching areas and make the UAVs cross the

intersection as quickly as possible. For this work, we assumed that the UAVs behave as instructed by the intersection manager and communication is assumed to be highly reliable.

To take into account the possible disturbances to the speed, we introduced a range of speed for the UAVs. We think that the speed range, to a certain extent, can reflect the effect of the variance of the actual speed. The speed range can easily be changed in our model and the effect of different speed ranges to the system's efficiency are shown in the simulation results. We think that the UAVs may move at the allowed speed range since current UAVs have the capability to follow the instructions very accurately.

For the model considered in this paper, we were able to show that our system works as expected. Through simulations, we were able to show that UAVs can cross the intersection efficiently. The efficiency is compared with the static optimal traffic light scheme to show that our system is highly efficient. However, when the intersection type or the number of layers changes, additional works are required. For example, we need to make changes on the allowed movements of the UAVs in the intersection.

We assumed in this work that there are no communication errors and delay is negligible. If the communication delay is significant or some messages (request or response) are lost, the associated UAVs may need to stop, which in effect will degrade the performance of the system. The effect of communication errors or messages lost on the system performance remains as a future work.

Another limitation of our work is that we assumed that UAVs could obtain their positions accurately. The position information of the UAV included in its reservation request needs to be accurate. Otherwise, the order of UAVs' arrivals may be incorrectly determined. To address this problem, we may need to add information in the request that specifies which UAV is ahead of which. We also need the accurate position of the UAV to correctly compute the time it enters the intersection. To consider the position inaccuracy in this issue, we may need to allow for certain errors in the position of the UAVs in scheduling them. Aside from the previously mentioned problems, if the UAV cannot obtain and provide accurate position information, it cannot behave in the approaching area as expected. This problem can be solved by increasing the lengths of the reservation zone and queueing zone appropriately. The effect of the inaccuracy in the position information to the performance of the system also remains as a future work.

Author Contributions: Conceptualization, M.C. and H.-H.C.; Formal analysis, A.R. and M.C.; Funding acquisition, M.C. and H.-H.C.; Methodology, A.R. and M.C.; Project administration, M.C. and H.-H.C.; Software, A.R. and H.-H.C.; Supervision, M.C.; Validation, A.R., M.C. and H.-H.C.; Writing—original draft, A.R.; Writing—review and editing, M.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2018R1D1A1B07049577 and NRF- 2017R1D1A1B03035324). This fund will also cover the costs to publish in open access.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Balamurugan, N.M.; Mohan, S.; Adimoolam, M.; John, A.; Wang, W. DOA tracking for seamless connectivity in beamformed IoT-based drones. *Comput. Stand. Interfaces* **2021**, *79*, 103564. [CrossRef]
2. Cai, Y.; Cui, F.; Shi, Q.; Zhao, M.; Li, G.Y. Dual-UAV-enabled secure communications: Joint trajectory design and user scheduling. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 1972–1985. [CrossRef]
3. Kopardekar, P.; Rios, J.; Prevot, T.; Johnson, M.; Jung, J.; Robinson, J.E. Unmanned aircraft system traffic management (UTM) concept of operations. In Proceedings of the AIAA Aviation and Aeronautics Forum, Washington, DC, USA, 13 June 2016.
4. Xu, C.; Liao, X.; Tan, J.; Ye, H.; Lu, H. Recent research progress of unmanned aerial vehicle regulation policies and technologies in urban low altitude. *IEEE Access* **2020**, *8*, 74175–74194. [CrossRef]
5. Dresner, K.; Stone, P. Multiagent traffic management: A reservation-based intersection control mechanism. In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, New York, NY, USA, 23 July 2004; pp. 530–537.

6. Kuru, K. Planning the Future of Smart Cities with Swarms of Fully Autonomous Unmanned Aerial Vehicles Using a Novel Framework. *IEEE Access* **2021**, *9*, 6571–6595. [CrossRef]
7. Ho, F.; Geraldes, R.; Gonçalves, A.; Rigault, B.; Sportich, B.; Kubo, D.; Cavazza, M.; Prendinger, H. Decentralized multi-agent path finding for UAV traffic management. *IEEE Trans. Intell. Transp. Syst.* **2020**, in press. [CrossRef]
8. Labib, N.S.; Danoy, G.; Musial, J.; Brust, M.R.; Bouvry, P. A multilayer low-altitude airspace model for UAV traffic management. In Proceedings of the 9th ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications, Miami, FL, USA, 25–29 November 2019; pp. 57–63.
9. Labib, N.S.; Danoy, G.; Musial, J.; Brust, M.R.; Bouvry, P. Internet of unmanned aerial vehicles—A multilayer low-altitude airspace model for distributed UAV traffic management. *Sensors* **2019**, *19*, 4779. [CrossRef] [PubMed]
10. Yasin, J.N.; Mohamed, S.A.; Haghbayan, M.H.; Heikkonen, J.; Tenhunen, H.; Plosila, J. Unmanned aerial vehicles (uavs): Collision avoidance systems and approaches. *IEEE Access* **2020**, *8*, 105139–105155. [CrossRef]
11. Yang, L.; Qi, J.; Xiao, J.; Yong, X. A literature review of UAV 3D path planning. In Proceedings of the 11th World Congress on Intelligent Control and Automation, Shenyang, China, 29 June 2014; pp. 2376–2381.
12. Lin, Y.; Saripalli, S. Sampling-based path planning for UAV collision avoidance. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 3179–3192. [CrossRef]
13. Lin, Y.; Saripalli, S. Path planning using 3D dubins curve for unmanned aerial vehicles. In Proceedings of the IEEE International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, FL, USA, 27–10 May 2014; pp. 296–304.
14. Hrabar, S. 3D path planning and stereo-based obstacle avoidance for rotorcraft UAVs. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 22 September 2008; pp. 807–814.
15. Yan, F.; Liu, Y.S.; Xiao, J.Z. Path planning in complex 3D environments using a probabilistic roadmap method. *Int. J. Control Autom.* **2013**, *10*, 525–533. [CrossRef]
16. Eun, Y.; Bang, H. Cooperative task assignment and path planning of multiple UAVs using genetic algorithm. In Proceedings of the AIAA Infotech at Aerospace 2007 Conference and Exhibit, Rohnert Park, CA, USA, 7–10 May 2007; p. 2982.
17. Lugo-Cárdenas, I.; Flores, G.; Salazar, S.; Lozano, R. Dubins path generation for a fixed wing UAV. In Proceedings of the International conference on unmanned aircraft systems (ICUAS), Orlando, FL, USA, 27 May 2014; pp. 339–346.
18. Yan, P.; Yan, Z.; Zheng, H.; Guo, J. A Fixed Wing UAV Path Planning Algorithm Based On Genetic Algorithm and Dubins Curve Theory. In Proceedings of the International Conference on Mechanical, Material and Aerospace Engineering, Wuhan, China, 10–13 May 2018.
19. Yan, F.; Zhu, X.; Zhou, Z.; Chu, J. A hierarchical mission planning method for simultaneous arrival of multi-UAV coalition. *Appl. Sci.* **2019**, *9*, 1986. [CrossRef]
20. Shah, M.A.; Aouf, N. 3D cooperative Pythagorean hodograph path planning and obstacle avoidance for multiple UAVs. In Proceedings of the IEEE 9th International Conference on Cybernetic Intelligent Systems, Reading, UK, 1 September 2010; pp. 1–6.
21. Nabi-Abdolyousefi, R.; Banazadeh, A. 3D offline path planning for a surveillance aerial vehicle using B-splines. In Proceedings of the International Conference on Advanced Mechatronic Systems, Luoyang, China, 25 September 2013; pp. 306–311.
22. Mittal, S.; Deb, K. Three-dimensional offline path planning for UAVs using multiobjective evolutionary algorithms. In Proceedings of the IEEE Congress on Evolutionary Computation, Singapore, 25 September 2007; pp. 3195–3202.
23. Nikolos, I.K.; Valavanis, K.P.; Tsourveloudis, N.C.; Kostaras, A.N. Evolutionary algorithm based offline/online path planner for UAV navigation. *IEEE Trans. Syst. Man Cybern. B* **2003**, *33*, 898–912. [CrossRef] [PubMed]
24. Zhang, X.; Chen, J.; Xin, B.; Fang, H. Online path planning for UAV using an improved differential evolution algorithm. *IFAC Proc. Vol.* **2011**, *44*, 6349–6354. [CrossRef]
25. Macharet, D.G.; Neto, A.A.; Campos, M.F. Feasible UAV path planning using genetic algorithms and Bézier curves. In Proceedings of the Brazilian Symposium on Artificial Intelligence, São Bernardo do Campo, Brazil, 23 October 2010; pp. 223–232.
26. Sathyaraj, B.M.; Jain, L.C.; Finn, A.; Drake, S. Multiple UAVs path planning algorithms: A comparative study. *Fuzzy Optim. Decis. Mak.* **2008**, *7*, 257–267. [CrossRef]
27. Meng, B.B.; Gao, X. UAV path planning based on bidirectional sparse A* search algorithm. In Proceedings of the International Conference on Intelligent Computation Technology and Automation, Changsha, China, 11 May 2010; pp. 1106–1109.
28. Zhan, W.; Wang, W.; Chen, N.; Wang, C. Efficient UAV path planning with multiconstraints in a 3D large battlefield environment. *Math. Probl. Eng.* **2014**, *2014*, 12. [CrossRef]
29. Koenig, S.; Likhachev, M. Fast replanning for navigation in unknown terrain. *IEEE Trans. Robot.* **2005**, *21*, 354–363. [CrossRef]
30. Shim, D.H.; Sastry, S. An evasive maneuvering algorithm for UAVs in see-and-avoid situations. In Proceedings of the IEEE American Control Conference, New York, NY, USA, 9–14 July 2007; pp. 3886–3891.
31. Ho, F.; Geraldes, R.; Goncalves, A.; Cavazza, M.; Prendinger, H. Improved conflict detection and resolution for service UAVs in shared airspace. *IEEE Trans. Veh. Technol.* **2018**, *68*, 1231–1242. [CrossRef]
32. Radmanesh, M.; Kumar, M. Flight formation of UAVs in presence of moving obstacles using fast-dynamic mixed integer linear programming. *Aerosp. Sci. Technol.* **2016**, *50*, 149–160. [CrossRef]
33. Selvam, P.K.; Raja, G.; Rajagopal, V.; Dev, K.; Knorr, S. Collision-free Path Planning for UAVs using Efficient Artificial Potential Field Algorithm. In Proceedings of the IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), Helsinki, Finland, 25–28 April 2021.

34. Hrabar, S. Reactive obstacle avoidance for rotorcraft uavs. In Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25 September 2011; pp. 4967–4974.
35. Sigurd, K.; How, J. UAV trajectory design using total field collision avoidance. In Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit, Austin, TX, USA, 11–14 August 2003; p. 5728.
36. Ferrera, E.; Alcántara, A.; Capitán, J.; Castaño, A.R.; Marrón, P.J.; Ollero, A. Decentralized 3d collision avoidance for multiple uavs in outdoor environments. *Sensors* **2018**, *18*, 4101. [CrossRef]
37. Choi, M.; Rubenecia, A.; Shon, T.; Choi, H.H. Velocity obstacle based 3d collision avoidance scheme for low-cost micro uavs. *Sustainability* **2017**, *9*, 1174. [CrossRef]
38. Huang, S.; Teo, R.S.H.; Liu, W. Distributed Cooperative Avoidance Control for Multi-Unmanned Aerial Vehicles. *Actuators* **2019**, *8*, 1. [CrossRef]
39. Albaker, B.M.; Rahim, N.A. Autonomous unmanned aircraft collision avoidance system based on geometric intersection. *Phys. Sci. Int. J.* **2011**, *6*, 391–401.
40. Douthwaite, J.A.; De Freitas, A.; Mihaylova, L.S. An interval approach to multiple unmanned aerial vehicle collision avoidance. In Proceedings of the 2017 Sensor Data Fusion: Trends, Solutions, Applications (SDF), Bonn, Germany, 10–12 October 2017.
41. Khosiawan, Y.; Park, Y.; Moon, I.; Nilakantan, J.M.; Nielsen, I. Task scheduling system for UAV operations in indoor environment. *Neural Comput. Appl.* **2019**, *31*, 5431–5459. [CrossRef]
42. Pérez-Carabaza, S.; Scherer, J.; Rinner, B.; López-Orozco, J.A.; Besada-Portas, E. UAV trajectory optimization for Minimum Time Search with communication constraints and collision avoidance. *Eng. Appl. Artif. Intel* **2019**, *85*, 357–371. [CrossRef]
43. Yang, Q.; Yoo, S.J. Optimal UAV path planning: Sensing data acquisition over IoT sensor networks using multi-objective bio-inspired algorithms. *IEEE Access* **2018**, *6*, 13671–13684. [CrossRef]
44. Bayrak, A.; Efe, M.Ö. FPGA based offline 3D UAV local path planner using evolutionary algorithms for unknown environments. In Proceedings of the Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, 23 October 2016; pp. 4778–4783.
45. Hayat, S.; Yanmaz, E.; Bettstetter, C.; Brown, T.X. Multi-objective drone path planning for search and rescue with quality-of-service requirements. *Auton Robot.* **2020**, *44*, 1183–1198. [CrossRef]
46. Wan, X.; Ghazzai, H.; Massoud, Y.; Menouar, H. Optimal collision-free navigation for multi-rotor UAV swarms in urban areas. In Proceedings of the 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), Kuala Lumpur, Malaysia, 28 April–1 May 2019.
47. Bahabry, A.; Wan, X.; Ghazzai, H.; Menouar, H.; Vesonder, G.; Massoud, Y. Low-altitude navigation for multi-rotor drones in urban areas. *IEEE Access* **2019**, *7*, 87716–87731. [CrossRef]
48. Bahabry, A.; Wan, X.; Ghazzai, H.; Vesonder, G.; Massoud, Y. Collision-free navigation and efficient scheduling for fleet of multi-rotor drones in smart city. In Proceedings of the 2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS), Dallas, TX, USA, 4–7 August 2019; pp. 552–555.
49. Suresh, P.; Saravanakumar, U.; Iwendi, C.; Mohan, S.; Srivastava, G. Field-programmable gate arrays in a low power vision system. *Comput. Electr. Eng.* **2021**, *90*, 106996.
50. Shakhatreh, H.; Sawalmeh, A.H.; Al-Fuqaha, A.; Dou, Z.; Almaita, E.; Khalil, I.; Othman, N.S.; Khreishah, A.; Guizani, M. Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges. *IEEE Access* **2019**, *7*, 48572–48634. [CrossRef]
51. Chen, X.; Tang, J.; Lao, S. Review of Unmanned Aerial Vehicle Swarm Communication Architectures and Routing Protocols. *Appl. Sci.* **2020**, *10*, 3661. [CrossRef]
52. Shiri, H.; Park, J.; Bennis, M. Communication-efficient massive UAV online path control: Federated learning meets mean-field game theory. *arXiv* **2020**, arXiv:2003.04451. [CrossRef]
53. He, L.; Bai, P.; Liang, X.; Zhang, J.; Wang, W. Feedback formation control of UAV swarm with multiple implicit leaders. *Aerosp. Sci. Technol.* **2018**, *72*, 327–334. [CrossRef]
54. Bai, G.; Li, Y.; Fang, Y.; Zhang, Y.A.; Tao, J. Network approach for resilience evaluation of a UAV swarm by considering communication limits. *Reliab. Eng. Syst. Saf.* **2020**, *193*, 106602. [CrossRef]
55. Mozaffari, M.; Saad, W.; Bennis, M.; Nam, Y.H.; Debbah, M. A tutorial on UAVs for wireless networks: Applications, challenges, and open problems. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2334–2360. [CrossRef]
56. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. Red-Black Trees. In *Introduction to Algorithms*, 3rd ed.; The MIT Press: Cambridge, MA, USA, 2009; pp. 308–323.