THE UNIVERSITY OF NOTTINGHAM

FACULTY OF ENGINEERING

SCHOOL OF ELECTRICAL AND ELECTRONICS ENGINEERING

THESIS (PhD)

HYBRIDIZED TRAJECTORY GENERATION AND MANY-OBJECTIVES OPTIMIZATION FOR MULTI-AGENT QUADROTOR UAVS

AUTHOR: PREMEELA THARUMANATHAN

FIRST SUPERVISOR: DR.HAIDER A. F. ALMURIB

SECOND SUPERVISOR: DR. NANDHA KUMAR

for my fantastic family, friends and supervisors.

CONTENTS

ABSTRACT

This research generates a large collection of optimized trajectories for multi-agent quadrotors. The hybridized algorithm extracts trajectories with various trade-off for all agents without discrimination. This allows the resources of all agents to contribute towards the completion of a task.

Two variations of multi-agent quadrotor missions are applied within this work. The first is spatially spread flight mission, MA-SPREAD whereas the second is formation flight, MA-FORMATION. The trajectories are designed within three environments: i) Highly Cluttered Indoor, ii) Cityscape and iii) Mountainous terrain. The initial path nodes are generated through a sampling based planner. Here, Rapidly Exploring Random Trees is expanded into Multi-Agent Rapidly Exploring Random Forest. These paths are used to form the initial population for Genetic Algorithm. Next, we apply Many-Objectives Optimization towards the optimization of all agents and its objectives.

This study strikes a balance between diverse and well minimized solutions through dimensionality reduction. Result shows that the algorithm can successfully find a diverse set of well minimized solutions within each environment. The end user will be supplied with high resolution visual imagery of each test environment and well-organized data that defines the trade-offs of each trajectory. These easy to understand information will assist the end user in making a final choice regarding the best multi-agent quadrotor trajectories for their mission.

LIST OF PUBLICATIONS

1. Premeela T.Nathan, Haider A. F. Almurib, Nandha T.Kumar, "Optimization of Nonlinearities through Control Techniques of the Quadrotor Aerial Vehicle", IEEE International Conference on Modeling, Simulation and Applied Optimization, ICMSAO 2011. (presented)

2. Premeela T. Nathan, Haider A. F. Almurib, Nandha T.Kumar, "A Review of Autonomous Multi-Agent Quad-rotor Control and Applications", IEEE International Conference on Mechatronics, ICOM 2011. (presented)

3. Haider A. F. Almurib, Premeela T. Nathan, Nandha T.Kumar, "Search and Rescue through Control Techniques of the Quadrotor Aerial Vehicle", IEEE International conference on Instrumentation, Control, Information Technology and System Integration, SICE 2011. (presented)

4. Premeela T. Nathan, Haider A. F. Almurib, Nandha T.Kumar, "Many-Objectives Optimization and Formation Flight Design for Quadrotors",Applied Soft Computing, 2016. (submitted)

CHAPTER 1: INTRODUCTION

This research designs a standardized platform that generates a large population of well minimized trajectories for multi-agent quadrotors. It is focused towards long distance trajectory planning across partially known environments. The quadrotors will fly across high-rise cityscapes, highly cluttered indoor environments and mountainous terrains. Initially, mapping the free space across the three-dimensional environments with a variety of challenges is prioritized. Next, a path planning algorithm that can generate paths for many agents simultaneously is designed. This work also aims to define objective functions that effectively distinguish the pros and cons of each trajectory. The values of these objective functions must be obtained through a stable multi-agent quadrotor control system. Lastly, this research aspires to design an algorithm that is capable of ranking and maintaining a diverse collection of trajectories across generations.

Here, we apply modified versions of Rapidly Exploring Random Trees, Genetic Algorithm and Dimensionality Reduced Many Objectives Dominance Optimization. Firstly, initial path waypoint generation and repair is performed through a multi-tree rapidly exploring random trees algorithm. Then, a modified genetic algorithm is designed to produce a diverse population of trajectories across all iterations. Thirdly, a standardized definition of the quadrotors' physical constraints and mission limitations are represented by a collection of objectives functions. The values of these objective functions are estimated through a multi-agent control system. Next, the collective optimization of a group of quadrotors for two different missions is performed through dimensionality reduced many-objectives optimization. The trajectories are ranked based on their objective values and diversity. In this research, both spread and adaptive formation flights are explored. With spatially spread flights, each agent's trajectory is designed independently. In this case, it is the combination of all the agents' independent trajectories that is optimized. For the second application, the trajectories of each agent and their designed formation shapes are optimized. This algorithm optimizes the multi-agent trajectories as a collective where the resources of all agents are equally important.

1.1 PROBLEM STATEMENT

Any path planner must consider the different challenges that can arise when planning and optimizing the trajectories of multiple agents simultaneously. Firstly, there must be access to a speedy control system that is used to predict the motion and state derivatives of each agent. Secondly, many works chose to generate the trajectories for each agent individually. A system that generates the best paths for all agents without sacrificing the performance of any agent is preferable. Thirdly, researchers have often chosen to focus on a few environments and objectives only. There is a necessity for a system that is adaptable to different types of terrains, tasks and objectives. A trajectory optimization algorithm that

can produce paths for a variety of terrains whilst considering many objectives simultaneously can be highly advantageous.

Next, end users are typically provided with a small collection of possible paths. There is a shortage of algorithms that can provide the end user with a large collection of diverse and well minimized trajectories. A large and diverse selection pool that contains 30 different options means that the end user has many possible choices. Lastly, any information regarding the trade-offs of each path can aid the end user in determining the best choice for their mission. Recent studies show that there is a lack of planners that do not require predetermined objective preferences. Also, many works do not provide full visual imagery or include in-depth knowledge regarding the benefits of each solution within the population. This research aims to fill these needs through the application of a multi-agent quadrotor trajectory generation and optimization algorithm.

1.2 OBJECTIVES

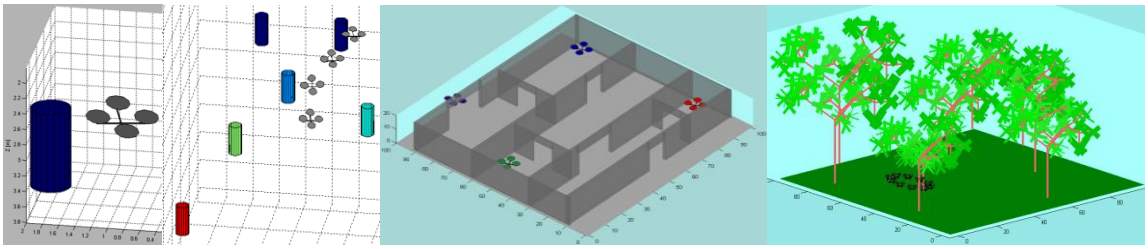
The objectives for this study is as stated below:

a) CONSTRUCT AND MAP FREE SPACE

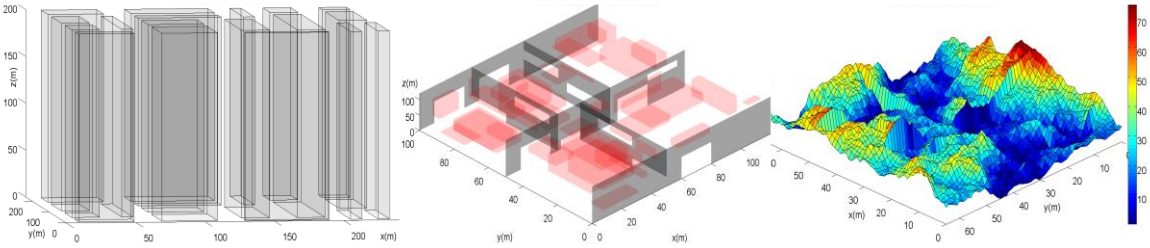
The first objective of this study is to construct and quickly identify the free space within three-dimensional test environments that mimic real-life flight challenges. The environments are not derived from real-life environments. The test spaces are designed to imitate the main structure of real-life environments. Thus, the simulated spaces can be randomly generated and look different for each experiment. Achieving this objective can be difficult because there needs to be a balance between accuracy and processing time. The path planner will be ready to design multi-agent quadrotor UAV trajectories for flight if the simulated test environments are similar to its real-life counterparts. On the other hand, a highly accurate environment requires a lot of detail within its space and this will increase the path planner's processing time. Finding a balance between these two conflicting goals is the key to achieving this objective.

The initial simulation environments that were used within this research project were focused on simple cylindrical obstacle shapes as shown in Figure 1.1(a). Here, a real-time simulated model of the quadrotor UAV was applied as well. This real-time model was viewable by the user while the algorithm is running. Thus, the image changes as the algorithm progresses. It contained the full mathematical model of the quadrotor within its PD control system. The rotational and translational movements of the quadrotor were visible on screen during the path planning process. Next, an indoor space that mirrors office lots and homes was introduced as can be seen in Figure 1.1(b). A forest test space as defined in Figure 1.1(c) was also created to mimic outdoor multi-agent flights. The issue with these

initial simulations of a high-resolution environments and multi-agent quadrotors is that it was computationally heavy. The path planning algorithm wasn't given full access to the processing system. This is because the real-time on-screen visuals of the many agents required a large portion of the processing power. It became obvious that these methods wouldn't work as the trajectory planning and optimization algorithm became more complex. Finally, an attempt to reach this objective is shown in Figure 1.1(d-f). Three highly challenging environments were designed. They are the cityscape, highly cluttered indoor space and mountainous terrain. The on-screen simulation was removed in favour of simple X shaped markers to define the movements of the quadrotors. The test environments were designed to be more accurate in definition as well. The end user can view these high-resolution images at the end of the simulation process as opposed to during path planning process.



Real-time simulation of (a) Quadrotor UAV. (b) Indoor flight of multi-agent quadrotors. (c) Formation flight across forestation.



High resolution imagery of (d) Cityscape. (e) Highly cluttered indoor space. (f) Mountainous terrain.

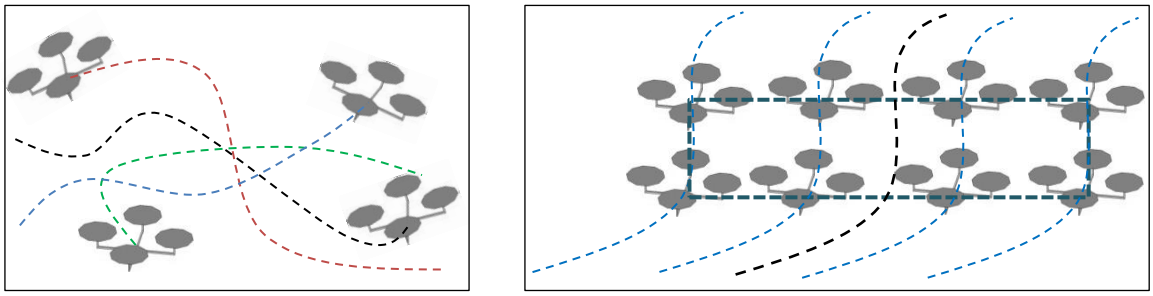
Fig. 1.1 Various MATLAB/SIMULINK simulated environments.

b) STANDARDIZE THE MISSIONS OF MULTI-AGENT QUADROTORS.

The secondary objective of this research project is to identify and generalize the various applications of multi-agent quadrotor UAVs. Achieving this target will allow the end user to apply the standardized platform towards a variety of multi-agent missions as opposed to just one. The challenging aspect of this objective is creating one path planning and optimization algorithm that caters to the needs of both multi-agent spatially spread and formation flight missions. These various multi-agent missions have been generalized into the MA-Spread and MA-Formation mission as shown in Figure 1.2(a-b).

The initial simulations within this study were performed with two very different algorithms. There was one for formation flights and another for basic spread flights. This approach proved to be extremely tedious because it required separate programs within

many subsections for both missions. It became clear that a singular algorithm for both missions was more cost and time effective. There are certain subsections within the standardized algorithm that can be kept simple for one mission but may be more complex to accommodate both missions. The MA-Spread mission requires independent flight trajectories for all agents whereas the MA-Formation only needs one formation reference trajectory. Similarly, the MA-Spread mission doesn't require close spaced flight structures whereas the MA-Formation does. This study attempts to achieve this objective by giving the end user the option of simplifying the algorithm. This can be accomplished by a simple change in the constant values of variables such as number of agents and type of mission. The MA-Spread algorithm designs paths for 4 agents that fly in different directions. The MA-Formation designs formation paths and shapes for 8 agents that fly near each other. Both mission use the same algorithm with one additional section for MA-Formation. Here, the algorithm is designed to automatically include the dynamic formation shape planner and its objectives if the MA-Formation mission is chosen.



(a) MA-Spread spatially spread multi-agent flight mission. (b) MA-Formation multi-agent formation flight mission.

Fig. 1.2 The two types of multi-agent quadrotor UAV missions.

c) PLACING SAFETY ZONES

Third objective for this study is to quickly and adequately define all the obstacles and their safety boundaries within each test environment. Achieving this objective requires the consideration of the same issues that were present in the first objective. The first challenge is to accurately define the height and shape changes within each obstacle. The second challenge is to identify the size of the safety boundary around each obstacle. This objective must be achieved with minimal processing time and complexity. Most importantly, the proper identification of the obstacles and its safety zones must produce multi-agent quadrotor UAV paths that have no collision points.

In the beginning, the algorithm was designed without placing safety zones around the obstacles as shown by the red path in Figure 1.3. This process can be used if the path's discrete nodes are not converted into smooth continuous trajectories for UAVs. On the other hand, the trajectories will collide with the sharp corners of the obstacles if minimal jerk splines are used. Next, standard sized safety zones were used for each test environments. This method achieves its objective but it causes the path planning algorithm

to design nondiverse paths. This is because each test space has different types of obstacles. Boundary sizes that are too large will limit the placement of path nodes across each test space. This will generate a collection of similar paths. The agents will collide with the many sharp corners of the buildings within the cityscape if the boundaries are too small. The narrow corridors of the indoor environment can close if the boundaries are too large as shown by the green path in Figure 1.3. Similarly, the small and gradual changes in height within the mountainous terrain aren't represented if the boundaries are too large. This research projects aims to achieve the third objective by applying different sized safety planes and boundaries across the three environments. The sizes of planes are defined based on the characteristics of each test space. This method will produce paths that are diverse and do not contain any collision points as shown by the blue path in Figure 1.3.

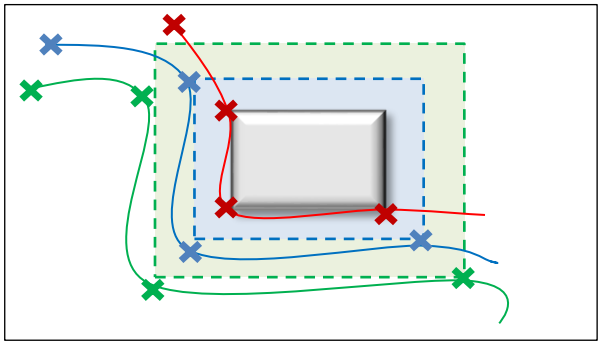


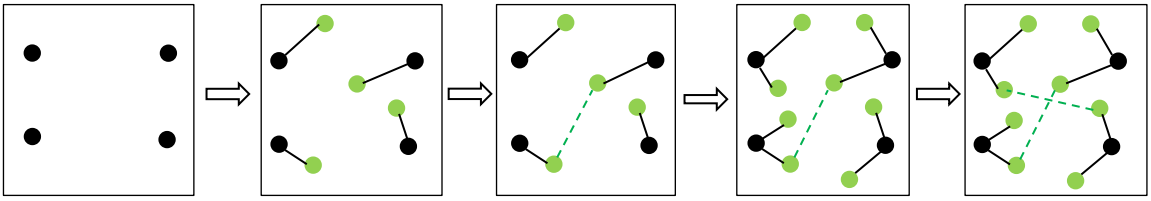
Fig. 1.3 The different sizes of safety zones around an obstacle.

d) GENERATE DIVERSE PATHS FOR THE INITIAL POPULATION.

The most important objective of any path planning algorithm is to generate highly diverse, collision free paths for the initial population of the optimization process. This study requires an offline, long range, multi-agent UAV path planner. This objective can be achieved in two steps. The first goal of the path planner is to fully explore the free space across three different test spaces. The second goal is to quickly extract a diverse and large collection of multi-agent paths. A few different path planners were applied throughout the course of this research project. The first was a basic shortest path algorithm such as Dijkstra's algorithm. Then, Virtual Potential Function (VPF) was used to plan paths for multiple agents. Next, consensus algorithm was implemented for multi-agent formation flight. These planners weren't fast enough or required a high amount of processing time. Some of the planners couldn't handle both the MA-Spread and MA-Formation missions.

Finally, a sampling-based planner was applied. This last choice showed a lot of potential in terms of achieving the objective of this study. A step-by-step diagram that describes the multi-agent sampling-based planner is presented in Figure 1.4 (a-e). The Rapidly Exploring Randomized Trees (RRT) free space mapping and path planning

algorithm begins by placing a root node (black) at the initial points of the multi-agents. Sample points (green) are then connected to the root node to form a tree branch (black). This study combines the trees through tree-to-tree linkages (green) to create a forest. The process is repeated until the environment is fully explored. To fully achieve this objective, a diverse collection of multi-agent paths must be quickly extracted. It is important that a well organised database is created. It is also crucial to implement a path filtration process to remove nondiverse paths from the tens of thousands of unique paths.



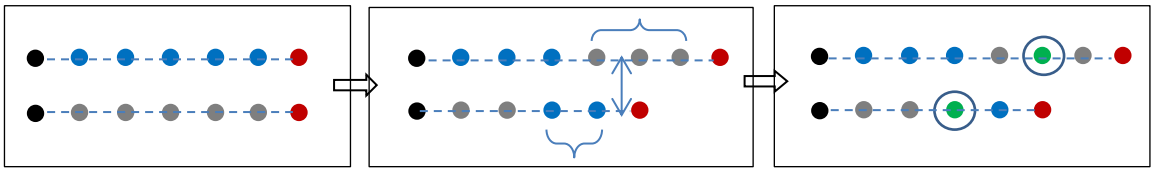
(a) Set the roots of each tree. (b) Connect sample points to the nearest tree. (c) Create forest links. (d) Further expand each tree. (e) Connect more forest links.

Fig. 1.4 The initial steps of the MA-RRF free space mapping and path planning algorithm.

e) UNIQUE POPULATION OF PATHS FOR EACH GENERATION.

The fifth objective is to create and maintain a diverse population of multi-agent paths for each generation. This objective can be split into a few goals. The first is to hybridize the MA-RRF paths to create new path populations for the optimization algorithm. The algorithm must be able to select suitable paths to merge together. It must also be able to create new collision free paths. The second goal is maintaining the level of path diversity across the many generations. This can be done by constantly creating new connections between path nodes.

The algorithm that will be able to reach these goals is Genetic Algorithm (GA). This planner can be applied as a refined path planner that uses the initial RRT path nodes as reference points. It is used to hybridize two paths to create two more new paths. It is also capable of exploring areas that are close in proximity to make minor improvements to these newly hybridized paths. Figure 1.5(a-c) shows the different stages that will be implemented within the GA. Figure 1.5(a) shows two parent paths that have the same initial and destination point. A crossover point is selected and the nodes within the two paths are switched as defined in Figure 1.5(b). Lastly, a singular node is chosen within these new paths as the mutation point. The coordinates of the nodes as shown in Figure 1.5(c) will be changed based on the type of mutation that is applied. These stages will allow the full path planning algorithm to achieve this objective.



(a) The two selected parent paths for the same agent. (b) Single point crossover. (c) External node mutation operator.

Fig. 1.5. The three types of operators that are applied within GA.

f) TRAJECTORIES THAT ARE SUITABLE FOR QUADROTORS.

Another objective of this research project is to design time-based, collision free, smooth trajectories that are suitable for multi-agent quadrotor UAVs. Smooth and minimal jerk trajectories are especially important for formation flights because it allows the agents to maintain their formation structure with ease. There are a few hurdles that need to be overcome to achieve this goal. The first is to design path nodes that have an appropriate distance from all obstacles. A smooth trajectory will follow the same direction of the path nodes but there will be some changes along certain subsections such as sharp corners. An example of this can be seen in Figure 1.6 where the red path defines the smooth spline that follows the initially planned grey path nodes.

The secondary challenge that occurs when trying to achieve this objective is finding a balance between maintaining and removing nodes across a path. Some nodes can be considered redundant because they do not add new information in terms of path direction. This can be tested by checking the difference in direction and curvature between three neighbouring nodes. Still, the maintenance of these nodes along the sharp corners of a path will create a smooth trajectory that is collision free. This objective can be achieved by using an adaptive method that only removes nodes across path sections that don't contain sharp bends. As shown in Figure 1.6, it also adds nodes to sections that contain aggressive turns with angles that are less than 90^{o}.

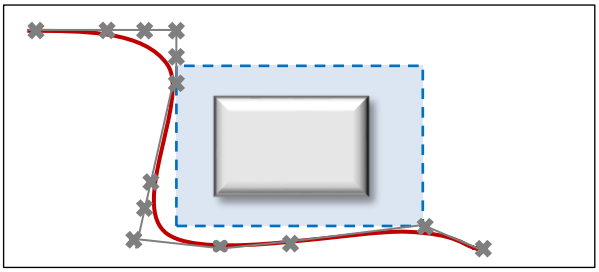


Fig. 1.6. The designed path nodes and its minimal jerk smooth spline.

g) PARALLEL RUN CLOSED-LOOP MULTI-AGENT CONTROL SYSTEM.

The seventh objective of this study is to model a noncomplex, parallel run, closed-loop multi-agent quadrotor UAV control system. The control system must overcome three

challenging stages to achieve this goal. The first stage is to implement a suitable mathematical model. This model will be designed to simulate the translational and rotational movements of the quadrotor UAV. Next, a fast and minimal error control system will be designed. This subsection will allow the control system to predict the path tracking error of the quadrotors as shown in Figure 1.7. The value of the error is important because it defines if a designed path can be followed by a real-life quadrotor. Lastly, the control system for the individual agent must be expanded into a parallel run multi-agent quadrotor control system.

The first control system that was designed for this research project was a Linear-quadratic regulator (LQR) controller with a full model of the quadrotor UAV. This control system can produce minimal control signal overshoot and error. The disadvantage of this controller was that it was complex in design. The secondary controller that was implemented was a PID control system alongside the nonsimplified mathematical model of the UAV. The control system that can achieve this objective is a more simplified version of the quadrotor UAV's mathematical model. This study will apply a PD controller with a simplified mathematical model for the control system. The system runs in series because the input of each subsection is dependent on the output of the previous subsection. It is fast and easily expandable into a multi-agent system.

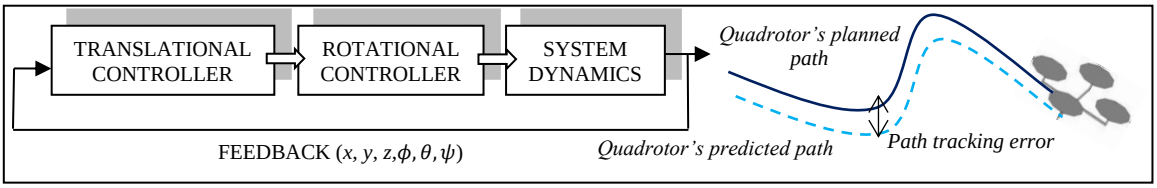


Fig. 1.7 The prediction system for the quadrotor UAV.

h) DEFINE A SET OF MULTI-AGENT QUADROTOR OBJECTIVES.

The next objective of this study is to mathematically and simplistically define a set of multi-agent quadrotor UAV mission-based objectives. The many objectives will be a collection of objective functions that have been applied with quadrotor UAVs. It considers objectives that consider a variety of missions, environments and sensors. Figure 1.8 shows an example of the objectives that are often considered within path planning algorithms. It can be challenging to convert real-life concerns into noncomplex mathematical equations. It is important that the equations are simplistic and easy to compute because they are used so often throughout the optimization algorithm. The objective functions do not have to be highly accurate since they exist to provide an indicator as to the trade-offs of each trajectory. A more accurate model can be applied in a smaller scale with a more modest sized population. This is out of the scope of this study since it only considers a large sized multi-agent path population.

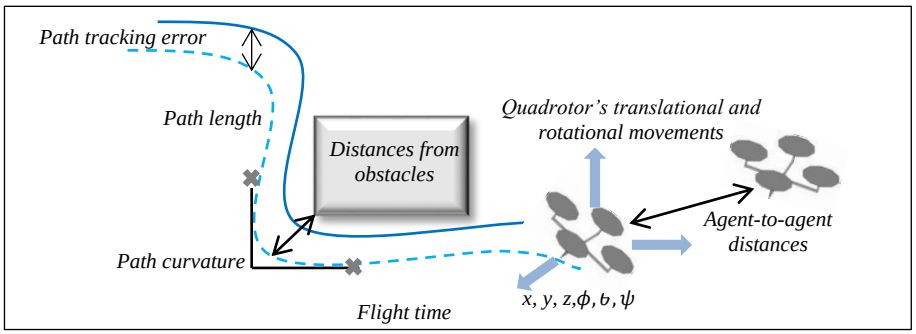


Fig. 1.8 The many different objectives that can be applied during a multi-agent quadrotor UAV mission.

i) WELL MINIMIZED YET DIVERSE FINAL SET OF TRAJECTORIES.

As previously defined, this research aims to minimize the values of the many objectives as opposed to finding the Pareto optimal solution set. Thus, the ninth objective of this study is to develop a well minimized and diverse final set of multi-agent trajectories for end users. There are two challenges that need to be overcome to achieve this objective. Firstly, it is important that the optimization algorithm can minimize the values of the many objective functions that are applied within this study. Secondly, the algorithm must be able to maintain a diverse set of trajectories so that the end user has a variety of multi-agent quadrotor trajectory options with various trade-offs in terms of objective values.

Initially, only a few objectives were considered. An aggregated function was used to minimize the values of these objectives simultaneously. Here, a basic multi-objective optimization algorithm was found to be sufficient. Later, the project was expanded to include both MA-Spread and MA-Formation missions. Both missions have their own costs and limitations. The algorithm needs to be able to minimize the values of many objectives at the same time. Thus, a many objectives optimization algorithm is implemented within this study. Dimensionality reduction and adaptive niching will be used to preserve well minimized solutions across generations. The progression of the Dimensionality Reduced Many-Objective Optimization (DRMOO) algorithm is shown in Figure 1.9.

This algorithm breaks the full objective set into smaller objective subsets. This is performed gradually by identifying objectives that do not conflict with the other objectives within its set. This is achieved by identifying objectives that do not contribute to the path ranking process within the optimization algorithm. Thus, the removal of a nonconflicting objective doesn't affect the rank of each path. These sets are then used in rotation for a constant number of generations. The rotation of both the full and smaller objective sets will allow the DRMOO algorithm to perform both local and global optimization simultaneously. The usage of the objective subsets will promote local optimization for the objectives within the current set. On the other hand, applying the full objective set promotes

global optimization for all objectives. This algorithm can achieve its objective of producing a diverse and well minimized set of multi-agent trajectories.

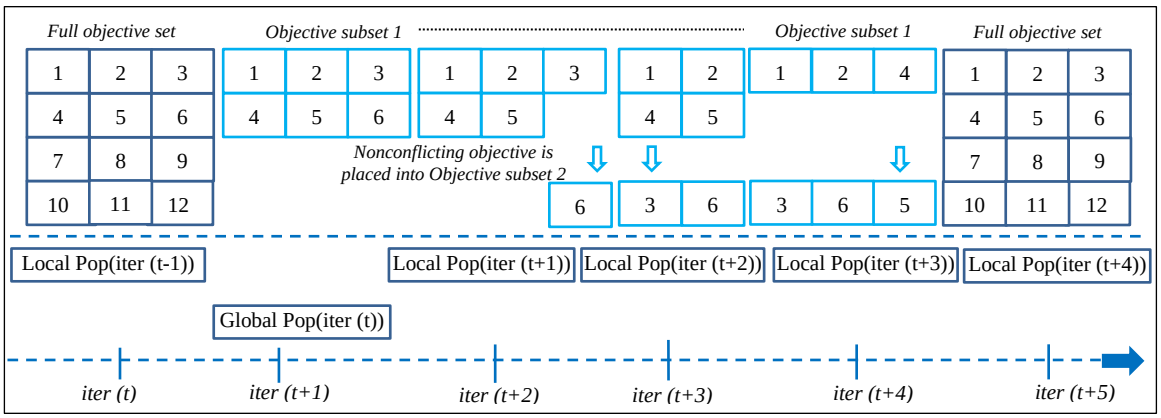


Fig. 1.9 The timeline of the DRMOO algorithm.

j) PROVIDE ORGANIZED AND UNDERSTANDABLE INFORMATION.

The final objective is to provide well organized and easily understandable information regarding the trade-offs of each multi-agent trajectory. This objective can be achieved in many ways. The end user may prefer a variety of multi-agent path options during the decision-making process. The most common way that researchers present their designed paths is through visual imagery. A high-resolution 3D image can be very useful because it allows the end user to rotate and zoom into the planned paths. Another popular method of analysing big amounts of data is through graphs. The changing values of the variables within the algorithm can be easily evaluated with graphs. Next, tables can be used to display and compare important data. Database management and visualization will allow the end user to view the results in an organized manner. These options will only present a small amount of information to the end user. Big data analysis software such as MATLAB/SIMULINK can be used to display all the data across many generations. This way, the end user has easy access to a large amount of information. The data must be stored within a database that is well organized. The achievement of this objective is possible with the implementation of these options.

1.3 CONTRIBUTIONS

The first contribution of this study is the development of a multi-agent RRT. A method for merging trees and extracting thousands of unique paths per agent is presented. Secondly, a modified GA is developed to produce a diverse population of trajectories for all agents at each generation. Another contribution of this study is its parallel run multi-agent quadrotor control system. The simulated movements of the quadrotor show if the planned paths are dynamically feasible, trackable and do not collide with obstacles. This work also presents a high-resolution formation planner. This leads to formation shapes that quickly adapt to

the changes within its environment. Next, each objective function is well defined to simulate real-life spread or formation flights. These objectives are applicable to many environments, quadrotor variations or sensory systems. Lastly, the DRMOO successfully finds a diverse set of well minimized trajectories for both scenarios. The results of this study show that all objectives are minimized or well-maintained without the extreme degradation of one objective over the other. The end user is presented with additional post processing knowledge and high-resolution imagery of each trajectory. This data can be easily understood and accessed when the end user attempts to choose the best path.

1.4 METHODOLOGY

This research performs trajectory planning and optimization for multi-agent quadrotors that is tested across two different applications. The algorithm is run on Windows 7, Intel ® Xeon (R) CPU E3-1230 V2, 3.30GHz, 8GB RAM, 64-bit operating system. Here, we apply modified versions of Rapidly Exploring Random Trees (RRT), Genetic Algorithm (GA) and Dimensionality Reduced Many Objectives Dominance Optimization (DRMOO). Each subsection that is shown in Figure 1.10 has been modified to suit and benefit from a multi-agent system. This combination produces an algorithm that creates a final population of diverse and well minimized trajectories for all quadrotors. This standardized platform is applied for both independent and formation flight missions. These trajectories are designed within three challenging environments which are highly cluttered indoor spaces, cityscapes and mountainous terrains. This study focuses on long distance trajectory planning across known environments. It can adapt to various terrains, tasks and objectives.

Firstly, a sampling based planner is applied to map the free space across each terrain. With a multi-agent system, multiple trees can be generated to further speed up the exploration process. The Multi-agent Rapidly Exploring Random Forest (MA-RRF) trajectory planner is designed to fully harness the advantages of having a multi-agent system. Initially, the start node of each agent is set as the root node of their RRT tree. The algorithm begins with separate trees and constantly checks for possible collision free linkages between them. The creation of the forest is done through the linking of open branches on different trees within the environment. Through this process, the individually rooted trees are quickly merged into a full forest. The tree branches and forest links must be efficiently stored in order to quickly extract the path nodes. A mutual database is constructed to store all the free space mapping information and tree branches from each agent. Tree branches that are within close ranges of any agent's goal node are also stored. This allows all agents to access the constantly updated shared database. This also increases efficiency and reduces the complexity of planning paths for many agents. The extraction of paths begins once the forest has fully explored the free space. Here, the algorithm extracts the many forest linkages that are formed between the different trees. The nodes for

each path are obtained across multiple tree subsections. This process produces a large collection of unique paths per agent. Finally, a filtration system is applied to remove paths that are similar in direction since the optimization process requires a diverse set of paths.

The many-objective optimization algorithm requires a population of thirty solutions at each generation. In this study, we apply GA for the generation of new trajectories at each generation. GA isn't used to generate the initial population because it requires more time to fully explore and extract feasible paths on its own. Thus, the path nodes that are designed by the MA-RRF algorithm will form the initial population for the optimization process. The basic GA requires additional modifications to create and maintain a collection of diverse paths across generations. This is extremely important because paths will converge towards similar directions especially within extremely constricted spaces. In this study, both the parent and child paths must produce nodes that don't exceed the similarity threshold to survive the selection process. Thus, GA will compare the paths within both generations. Next, single point crossover and three types of mutation are applied. In most cases, the newly generated child path is filled with collision points after undergoing the crossover and mutation process. Due to this, MA-RRF is applied to perform speedy path repair. Lastly, four additional post processing operators are applied. Any repetitive loops that are within the child paths are removed. Child paths that hold too few nodes for spline creation are padded with additional nodes. More nodes are also generated across the path sections with turns that are smaller than 90 degrees. Lastly, a similarity test is performed to compare the new child path to the current population. If the offspring is collision free and passes through these post processing steps successfully, it is stored as a member of the next generation.

The new generation of multi-agent paths will be converted into trajectories through the application of fifth order splines. These smooth trajectories are used as input for the parallel run multi-agent control system. This system is a combination of the control system and mathematical model of each agent. The control system executes parallel simulation for all agents on a multi-thread processing system. The estimated translational and rotational movement of the quadrotors will then be used to estimate the values of the many objective functions. In this study, twelve objective functions are defined for multi-agent spread (MA-Spread) mission. Eight are standard objectives whereas four objectives are specific to the MA-Spread mission. The standard objectives are shown in Section 4.1. The objectives that are specifically designed for spatially spread flight is shown in Section 4.2.3. With MA-Spread, thirty randomized combinations of four agents' flight paths are optimized for each generation. Each generation produces a new set of path combinations. These paths are a mesh of parent paths that are maintained across generations and newly planned paths. This method ensures that the resources of all agents are taken into consideration at each stage of the algorithm.

The formation flight (MA-Formation) missions also apply twelve objective functions. Here as well, eight are standard objectives and four are specific to formation flight missions. The standard objectives are shown in Section 4.1. The objectives that are specifically designed for formation flight is shown in Section 4.3.4. With the MA-Formation application, thirty formation reference paths are designed at each generation. These formation reference trajectory nodes are applied as the reference coordinates for the dynamic formation planner. Here, the formation planner works in high resolution. This level of resolution is advantageous for full obstacle clearance and the design of adaptive formation shapes. Both the multi-agent trajectories and its formation designs are applied towards determining the values of the many objective functions.

Lastly, this study applies DRMOO to generate smaller objective subsets. DRMOO promotes the creation of subsets that contain three or more conflicting objective functions. The objective subsets are created by comparing the number of nondominated solutions that remain within a population before and after the removal of an objective function. The objective function is found to be nonconflicting if the number of nondominated solutions remains similar despite its removal. This is because nonconflicting objectives make minimal contributions to the ranking process. The objective function is removed from its subset, placed within another subset and retested for redundancy as the iterations progress. As the algorithm progresses, these objective subsets are used in rotation. The full objective set with twelve objectives is reintroduced at the end of each interval. The application of both the full objective set and subsets allow the algorithm to perform both local and global optimization simultaneously. Also, no objectives are fully eliminated. This is advantageous in cases where an error has been made in determining nonconflicting objectives. At each generation, the solutions are ranked based on the current objective set. Next, adaptive niching is performed on the remaining population to determine which are most diverse. The average distances between all solutions are used to determine the current niche radius. This process encourages the degradation of crowded solutions and the enhancement of cluster representative solutions. Both the trajectory population's ranking and niching process collectively maintain solutions that are well minimized and diverse.

This hybridized trajectory planning algorithm is a mesh of MA-RRF, GA and DRMOO. It successfully generates a large collection of trajectories for multiple quadrotors simultaneously. Here, all objectives are minimized or maintained without the extreme degradation of one objective over the other. The end user is delivered easily interpretable knowledge to make a final decision. It has optimized a team of quadrotors collectively as opposed to individually. The path planner utilizes algorithms have the advantage of parallel processing. The control system, sampling based planner, genetic algorithm and many objectives optimization are designed to run on any multi thread system.

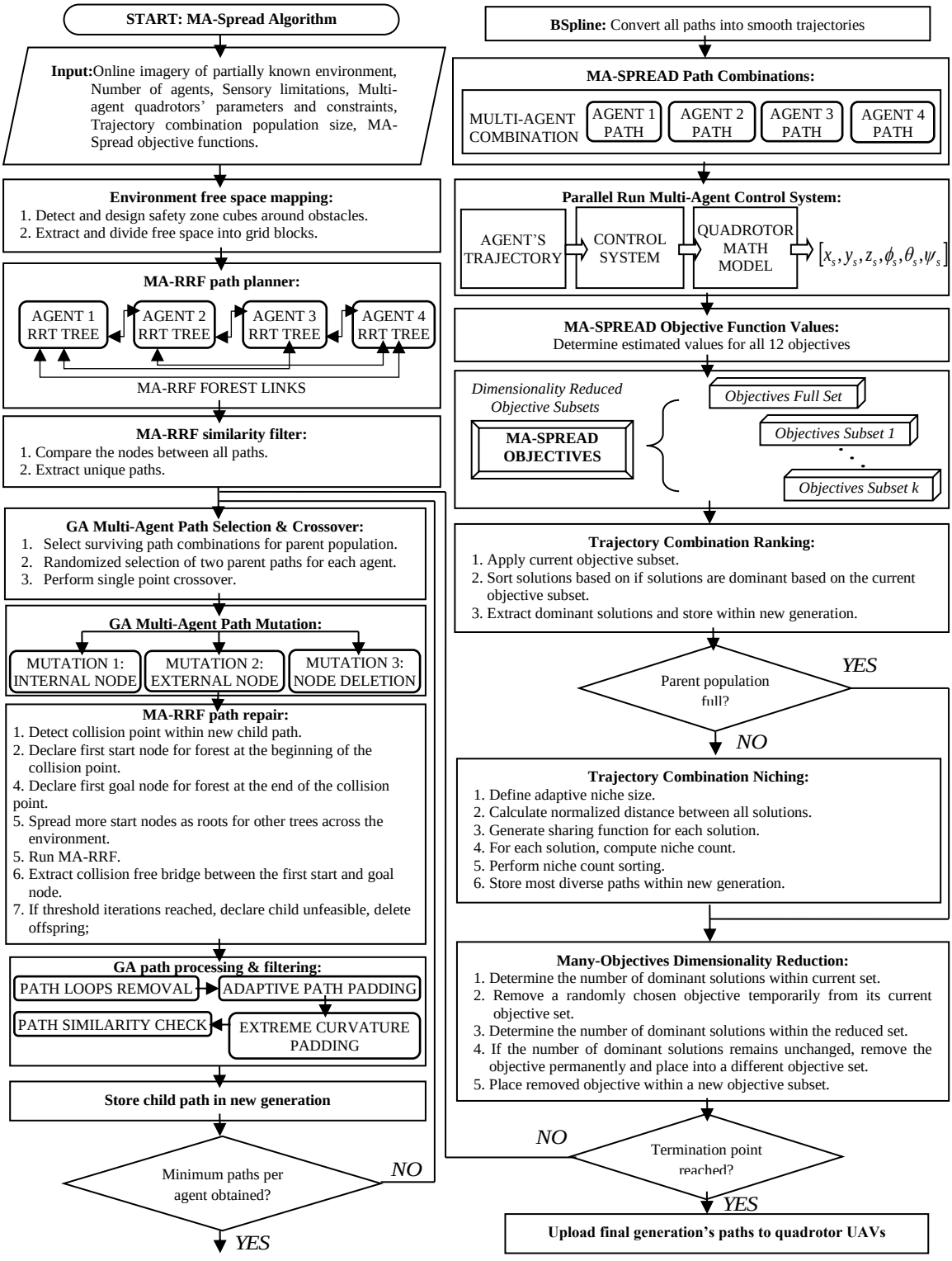


Fig. 1.10 Flow chart of MA-SPREAD trajectory planning algorithm.

1.5 THESIS OUTLINE

This thesis comprises of two main sections. The first half of the thesis lays the groundwork for the full multi-agent quadrotor UAV path planning and optimization algorithm. It shows the reader the different algorithms such as MA-RRF, GA and DRMOO that are merged to create the entire trajectory planning algorithm. These chapters will define the various theories within each algorithm and the reason they were chosen. It also defined the structure of each algorithm through mathematical equations. The reader will also be able to view the different missions, objectives and environments that are applied in this thesis. The second part of this thesis shows the results of the multi-agent path planning and optimization algorithm. Here, the best trajectories for each mission are presented in a variety of forms. The data that is derived from the final population of trajectories is displayed in graphs, tables and high-resolution imageries. These two subsections of the thesis will collectively show the theory, results and analysis of the unified algorithm.

Chapter 2 describes the different literatures that inspired the direction that was taken in this research project. It discusses and analyses the relevant research that have been produced by previous researchers, start-ups and large corporations. This chapter begins by describing the history of the quadrotor and the physical modifications that have been performed across the years. It also shows that the quadrotor has been applied within many challenging environments. Next, the chapter proceeds to define the different path planning algorithms that have been used for both the spatially spread and formation flight missions. This section leads to the analysis of the many objectives and limitations that are often applied within these two missions. Lastly, this chapter shows the various optimization methods that have been applied with multi-agent systems. It also defines the different types of many-objectives optimization algorithms that have been published. The chapter closes by analysing a specific subsection of many-objectives optimization which involves dimensionality reduction.

Then, Chapter 3 describes the theory and structure of the two hybridized path planners that are applied within this study. It describes the creation of the multi-agent quadrotor UAVs' hybridized paths by both the MA-RRF and GA planners. This chapter starts by illustrating the three test spaces that are applied within this study. Next, the framework of the MA-RRF planner is presented. Here, the MA-RRF planner acts as the initial path planner. This section shows the reader the free space sampling and mapping process. The results display the initial path population that has been designed for the MA-Spread and MA-Formation missions. Next, GA acts as a more refined path planner that hybridizes the nodes of the MA-RRF paths to create new generations of trajectories. This chapter continues by defining the different operators and post processing procedures that are used within GA. Finally, the basic structure of the quadrotor UAV's mathematical model and control system is defined. The reader is shown how these sections collectively

form a multi-agent quadrotor control system. This control system defines the translational and rotational movements of an agent. Thus, the data gives the reader an insight as to the trajectories that can be tracked by a real-life agent.

Chapter 4 describes the many objectives that are applied within the MA-Spread and MA-Formation missions. It also shows how these objectives are used within the DRMOO optimization algorithm. This chapter begins by defining the standardized objectives that are used for both missions. The next section defines the MA-Spread mission and its specific objectives. Here, the reader can see that the independent trajectories of four agents are optimized as a collective team. This part of the thesis also defines the estimated data that flows across the many parts of the algorithm when it is applied within the MA-Spread mission. Then, this chapter describes the MA-Formation mission. Similar to the prior subsection, the objectives that are used only for the MA-Formation missions are also mathematically defined. In this case, the formation reference trajectory, its shapes and the independent trajectories of eight agents are optimized simultaneously. The variables that are applied within the optimization process are defined by a high-resolution formation planner. Lastly, the structure of the DRMOO many-objectives optimization algorithm is presented within this chapter. This section shows the reader the different concepts that are used within the DRMOO algorithm such as dimensionality reduction and adaptive niching. Thus, the reader can understand the reasons why the DRMOO is suitable for optimizing a large collection of objectives and multi-agent trajectories.

This thesis presents the results of the multi-agent quadrotor UAV trajectory planning and optimization algorithm. The results for the two missions across three test environments are shown in Chapter 5. Each section begins by showing the changing values of the number of dominant solutions and the level of diversity across generations. The next part of each section shows the results of the final trajectory population. It allows the reader to analyse the level of optimization that has occurred from the first to final generation. The results and its data are displayed in table form and high-resolution imagery. Each section closes with detailed data regarding the objective values for each solution within the trajectory population. This part defines the pros and cons of each option to the reader. Finally, the conclusion of this study is presented in Chapter 6. This chapter analyses the implications, limitations and contributions of this research project.

CHAPTER 2: LITERATURE REVIEW

The design for a trajectory planner of multi-agent UAVs is heavily dependent on its real-life application. Today, most users still prefer remote controlled multi-rotor aerial vehicles. Still, unmanned versions are gaining popularity amongst elite users. This elite group comprises of users with access to funds and current research. Thus, most UAV systems are designed for governmental bodies, large corporations, research facilities and entrepreneurs. These end users will be applying the UAVs towards many different applications. The UAV that is most commonly used is the quadrotor. The four-rotor aerial vehicle is easily expandable to form a multi-agent system. Implementing a cooperative multi-agent system means that there is potential that missions can be completed at a much faster rate. Multi-agent quadrotors are frequently used for collective missions such as search and rescue or reconnaissance. The quadrotors can also be applied for target tracking as well as forming ad hoc wireless networks. Delivery companies are also attempting to transport light weight packages across urban environments. The most popular application of the quadrotor is for the creation of media content. Quadrotors are being used by journalists and scientists to capture high definition videos. Here, the various sensory systems that operate simultaneously can efficiently collect data for terrain mapping and wildlife research. Similarly, many hobbyists apply the quadrotor for capturing photographs of important family events.

Some of these applications are simplistic and only require a path planning algorithm that is easy to implement. On the other hand, more complex missions will benefit from a well-designed trajectory planning algorithm. A multi-agent system can be challenging because each agent within a team can be an asset or a liability. A path planning algorithm must be capable of harnessing the resources of each agent within its team. This research generates a large collection of optimized trajectories for multi-agent quadrotors. It can adapt to all types of terrains, tasks and objectives. This chapter presents background information on the various algorithms that form the final trajectory planning algorithm. Firstly, the history of the quadrotor aerial vehicle is presented. Here, we define the quadrotor's physical changes through time. This section also describes the various environments that the multi-agent quadrotors fly across. Next, the many objectives that are typically applied within different quadrotor missions are discussed. The two applications that are highlighted within this research are multi-agent spread and formation flights.

The next section defines the different trajectory planning algorithms that have been used for aerial vehicles. In this study, focus is placed upon sampling based planners such as rapidly random exploring trees. Lastly, the various algorithms that are used for trajectory optimization is presented. Researches that apply multi-objective optimization towards path planning are explored. In this work, emphasis is placed upon optimization through genetic algorithm. The concepts that are introduced in multi-objective optimization algorithms are used to optimize many objectives as well. This

section also shows the different methods that are used to improve the many-objectives optimization process. The studies that are presented within this literature review are used as the building blocks for this research. All three subsections are improved upon and hybridized to form the final multi-agent quadrotor trajectory planning algorithm.

2.1 QUADROTOR AERIAL VEHICLES

These days, there are many variations of the multi-rotor aerial vehicle. Each variation is typically named based on the number of rotors that are attached to the vehicle. The most commonly available multi-rotor system is the quadrotor or quadcopter. It is an aerial vehicle that has four rotors that is easily designed and assembled. It is typically made with durable materials such as carbon fibre and high-resistance plastic.

The quadrotor can be used for both small and large scale cooperative flight missions. This vehicle is often used in both research facilities and businesses. Thus, its commercial popularity has caused a drop in its manufacturing costs. Similarly, it is manufactured across many countries and can be easily purchased online or off-the-shelf. Quadrotors are often equipped with many types of sensory systems that can transmit real-time data. Progresses in measurement units and electronics have also produced an increase in the data processing capabilities of UAVs. The quadrotor is also capable of vertical take-off and landing as well as highly aggressive manoeuvres. Many users have chosen to use multi-rotor systems as opposed to fixed wing vehicles due to its high level of stability. The surging popularity of the quadrotor within the consumer market is due to its ability to remain stable whilst delivering clear imagery in real-time. It can undertake aggressive turns and capture videos at high definition. The fast evolution of the quadrotor and its software shows that it can be easily expandable to a large sized multi-agent system. This study aims to fully utilize the advantages of using the quadrotor for multi-agent missions.

This research has chosen to plan optimal trajectories for the quadrotor UAV within complex three-dimensional environments. In comparison to fixed wing and other rotary winged UAVs, the choice of applying the multi-agent platform to quadrotors is due to its benefits outweighing its disadvantages as follows,

- *Good agility in missions that require high manoeuvrability*. The quadrotor is highly manoeuvrable. Highly aggressive turns are performed smoothly with minimal obstacle buffer region.

- *Increase in payload capacity*. The quadrotor creates more lift thrust than conventional helicopters. Therefore, it can lift higher payloads. Multi-agent quadrotors offer more lift as a collective and can manage heavier weights.

- *Performs Vertical Take Off and Landing (VTOL)*. Spinning directions of the rotors are set to balance the moments. The balance that is achieved by the four

rotors eliminates the need for a tail rotor. Thus, the quadrotor can hover above targets unlike fixed wing UAVs. This reduces fuel consumption.

- *Hard to reach areas made easily accessible*. The quadrotor is capable of manoeuvring across narrow passages as well as highly cluttered spaces.

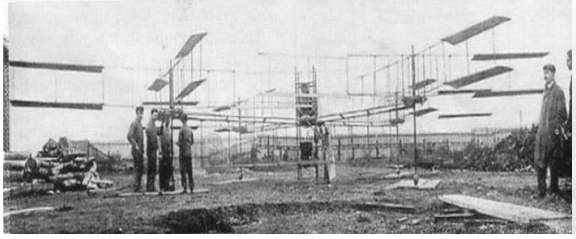
- *Cost effective and simple to build*. It offers a great platform for autonomous unmanned aerial vehicle research projects. It can be easily obtained through online shopping or off the shelf at malls. It is cheap enough for hobbyists as well.

- *Risks to humans are reduced*. The quadrotor is suitable for applications such as investigations, rescue missions and film making.

This section explores three types of quadrotor variations which are physical structure miniaturization, environment based hybridization and passenger transportation. These variations show that the quadrotor can be modified to complete any application or move across any environment.

2.1.1 QUADROTOR DEVELOPMENTS

The initial structure of the quadrotor or quadcopter began with the design of rotary winged Gyroplane No.1. The aerial vehicle was designed in year 1907 by Louis Breguet, Jacques Breguet and assisted by Professor Charles Richet [1]. Figure 2.1(a) shows a minimalistic structure that is made from steel. The corners of the Gyroplane No.1. are attached with rotors that are stacked upon each other. This basic design was further improved in 1922 by Georges de Bothezat as shown in Figure 2.1(b). His helicopter closely resembles the cross-frame physicality of the modern quadrotor. This design has six wide blades at the end of its four arms [2]. Both manually controlled designs were heavy and capable of short flights at low altitudes. Tests show that these vehicles were unstable due to the lack of a proper control or landing system. These quadcopters would have required high costs of production. Also, the energy that was being supplied to the vehicles was insufficient. Though imperfect, these models have inspired the development of today's helicopter. Thus, the idea of placing multiple rotors to produce lift and aggressively manoeuvre across test spaces has inspired countless variations of multi-rotor vehicles designs.



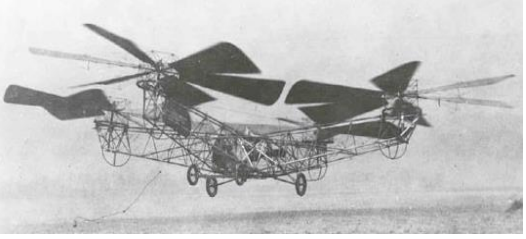


Fig. 2.1. (a) Gyroplane No.1 [1] (b) Georges de Bothezat's helicopter [2]

The design of the Gyroplane No.1 is large and cannot be flown across smaller spaces. A reduction in size is necessary to use these unmanned vehicles across constrained spaces. One of the most popular quadrotor that is low cost is the Parrot AR. Drone [3-4]. It comes in a variety of colours, multiple real-time games and live video feed. It is 28 x 28 inches in size and 13.4 ounces in weight when its hull is attached. The second version of the drone is controlled by a smart phone and comes fully equipped with WiFi network. The quadrotors are also produced with a high definition camera. Spare parts are repairable or purchasable online. Thus, the average user can easily assemble and fly the Parrot AR. Drone.

As shown in Figure 2.2, some designers have chosen to further reduce the size of the quadrotor. The Parrot Minidrone Rolling Spider that weighs 55 grams was released in August 2014 [5]. It is also capable of connecting to both the Android and Apple tablet's operating software. The Lil' Draganflyer Nano Quadrotor is extremely small. It is and built for indoor or outdoor flights that have minimal amount of wind resistance [6]. These miniature drones are capable of environment mapping across workspaces and homes. It is the ideal tool for surveillance that requires the drone to fly over and under clutter. The probability of damage towards civilians, pets and home goods is minimal. More importantly, it also fills the need for UAVs required by academic or research facilities. There are less dangerous and easier to control within tight spaces as compared to a full sized, high speed quadrotor. Here, smaller scaled versions of test environments can be constructed indoors for experiments. Both the control and navigation of multi-agent UAVs can be performed safely. The information that is obtained from these miniature versions can be highly valuable before attempting to fly larger drones across more dangerous locations.

Fig. 2.2. (a) Parrot Minidrone Rolling Spider [4] (b) Lil' Draganflyer Nano Quadrotor [6]

Another form of structure manipulation is the hybridization of ground and aerial motion. Samples of these designs are shown in Figure 2.3. The B is designed with the traditional wheeled vehicle in mind except each wheel contains a rotor [7]. It is a hybridization of the remote-controlled car and the quadrotor. The benefit of this design is its ability to easily switch between flight and on the ground driving. Similarly, the Hybrid Exploration Robot for Air and Land Deployment (HERALD) integrates legs with wheels. This hybridization allows the HERALD's to have seven degrees of freedom [8]. Besides wheeled designs, avian inspired graspers have also been attached to the quadrotor. This allows the UAV to perch and conserve fuel when necessary [9]. These quadrotors are capable of navigating across many types of terrains such as

disaster sites or mountainous terrains. In these cases, it can be dangerous for humans to transport the quadrotors towards its take-off location. It is advantageous for the quadrotor to be able to move on ground without human interference.

Fig. 2.3. (a) The B [7] (b) HERALD [8] (c) Avian inspired graspers on a quadrotor [9]

Recent developments show that there is an interest in using the quadrotor for transporting passengers or payloads. In this case, a trajectory planner must be capable of generating paths that consider the common direction of all agents. Most studies focus on payload transportation by flying in formation. As shown in Figure 2.4, there are some works that have modified the physical structure of the quadrotor. The first manned multicopter system was successfully flown by E-Volo in 2011 [10]. Here, the maximum payload of the quadrotor is enhanced by connecting a few agents across a singular structure. Figure 2.4(a) shows a structure that connects four quadrotors to carry more payloads. In this experiment, a human passenger is successfully lifted off the ground. Another variation of a manned quadrotor is the Hoverbike helicopter [11]. This model has combined the structure of a motorbike with the rotors of an aerial vehicle. The difference between the Hoverbike and the typical quadrotor design is the overlapping of two rotors to reduce size. The design comes with the option for manned or unmanned flight with a maximum of 270kg take-off weight. Whilst a commercial model isn't currently for sale, the rapid progression of the quadrotor shows that it can be used as a privately owned aerial vehicle in the future.

The physical variations that have been described show that the quadrotor can easily be modified. Thus, these agents can be used for a variety of applications. The aerial vehicle is typically flown across locations that can be hazardous for humans. This is because there can be extreme and sudden changes in terrain height. Besides understanding the physical structure of the quadrotor, it is also important that the designer of a trajectory planning algorithm considers all possible environments.

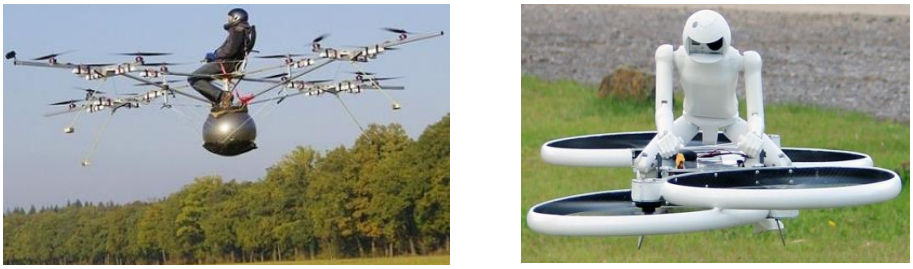


Fig. 2.4. (a) E-Volo's first manned multicopter [10] (b) Hoverbike [11]

2.1.2 QUADROTOR FLIGHTS ACROSS VARIOUS ENVIRONMENTS

The quadrotor can fly across many terrains that have different weather conditions. A path planner that can adapt to a variety of environments is highly advantageous for the end user. Preliminary knowledge on the type of environments and its challenges can be beneficial for the designer of the algorithm. Prior studies that experiment across both natural and manmade spaces can provide insight in terms of mapping, obstacle detection, safety and fuel consumption. This section explores outdoor environments with different weather conditions such as mountainous terrains, seas, forests, ice-covered landmasses and volcanoes. It also analyses manmade buildings with high amounts of clutter such as cityscapes, offices and residential areas. The information that is obtained from these studies will provide guidance for the development of a standardized trajectory planner that can accommodate the variations within these terrains.

Aerial vehicles are most commonly used for surveillance and photography. This is performed through high definition image and video capturing. The application of the quadrotor towards real-time image capturing allows humans to define unknown areas across the world. Future research will be aided by the data that is collected from these rotary vehicles. Deep sea exploration has proven to be an extremely challenging task for human divers. In the past year, there have been researchers that attempt to fly the quadrotor over and into seas. Researchers at Rutgers University have developed a quadrotor that can travel across air and manoeuvre underwater [12]. Figure 2.5(a) shows that the quadrotor can successfully transition between flying above and under a pool of water. The drone can perform low speed vertical and horizontal motions underwater. The quadrotor, Pars is being developed by The RTS Lab for search and rescue of potential drowning victims [13]. The drone is shown in Figure 2.5(b). Initially, the drone uses a ship as a base station. The drone takes off from its base station and flies over the sea to locate missing victims. The quadrotor is powered by solar energy and performs localization through satellite data. These drones will be able to transmit data such as wind speed, water turbulence and sea levels. The agents will provide imagery of different sea creatures and their habitats. It also allows the user to detect traces of hazardous pollution as well as ship wreckages. One advantage of sea exploration is the lack of obstacles that exist within the agent's path. The pathway is clear and path planning can be easily executed.

Fig. 2.5. (a) Rutgers University underwater drone [12]

(b) Rescue drone, Pars by RTS Lab [13]

Another popular test space is outdoor locations such as forestation and mountainous terrains. For geographical explorers, the UAV is used to map mountainous terrains at high altitudes. Mountainous terrains can be unsafe for humans to explore without prior information. Here, the unmanned vehicle proves to be an asset in determining the oxygen density at high altitudes or identifying sudden drops across the terrain. Microdrones GmbH has created a quadrotor that can fly at high altitudes [14]. As shown in Figure 2.6(a), the microdrone can fly across the Alps whilst sending real-time imagery. The path planner had to quickly replan certain sections of the trajectory due to an error in determining amount of the snow across the mountain peaks. The multi-agent unmanned system as shown in Figure 2.6(b) is often used across forests to collect a variety of data. Information such as animal population, the variance of fruits, plant health, environmental pollution and the water levels across the soil can be easily obtained. Forests are built with obstacles of varied heights in the form of trees that produce either compact or spread growth of leaves. The quadrotor can handle these challenges given its small size and aggressive manoeuvring.

Path planning outdoors can be complex and challenging. In [15], real-time generation of waypoints for a quadrotor is performed within a forest environment. Here, onboard motion estimation and path planning are implemented within a small forest with sparse trees. Mixed-integer optimization is performed in [16] where an obstacle free outdoor space is divided into convex regions. Adapting this algorithm towards more cluttered environments will be challenging since the planner neatly divides the spaces into large convex regions. Study [17] implements a low-resolution visibility-graph across an outdoor test space that contains restricted airspaces. The quadrotor's path planner is only capable of avoiding obstacles of similar height. Expanding these algorithms towards a more cluttered environment would require a higher sampling rate. Most algorithms are designed to perform in a singular type of environment. An adaptive trajectory planning algorithm is necessary for the quadrotors to successfully complete tasks in different environments that hold different sets of complexities.

Fig. 2.6. (a) Quadrotor flight across high altitude environments [14] (b) Navigating around forestation [18]

Another location that limits human exploration is environments with extreme temperatures or atmosphere toxicity. Here, the quadrotor performs better than humans. As visible in Figure 2.7, the agents can collect data across unexplored places such as the Antarctica. A DJI Phantom 2 quadrotor that was fitted with a GoPro Hero3 action

camera perished whilst flying into Iceland's Bardarbunga volcano [19]. Despite self-destruction, the quadrotor could capture unseen views of the core of the volcano in real-time. Similarly, Tohoku University in Japan is developing an unmanned aerial that collaborates with ground robots for cases of sudden volcanic eruptions that are happening frequently in the country [20]. Forest fires are a common occurrence in Australia. Drones can provide a view of the damage caused by the fire. This allows the authorities to quickly evacuate any residential areas that are nearby. It is also highly risky for humans to be around war zones. Today, the military system is slowly replacing manned vehicles with unmanned options. K-MAX dual rotor robocopters were successfully deployed across Afghanistan for autonomous cargo delivery that weighs over 750 pounds [21]. These unmanned aerial robots are advantageous because they are less prone to human error or causalities if it crashes.

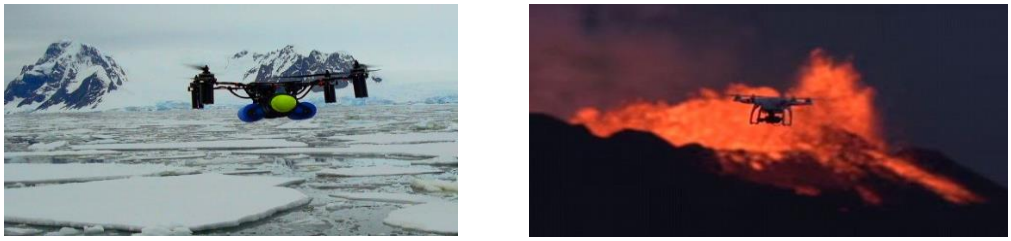


Fig. 2.7. Quadrotor flight across sea ice and volcanoes [19-20]

The most common test space for the commercial quadrotors is manmade structures such as indoor environments or cityscapes. Cityscapes pose a threat in terms of narrow passages with extreme bends. Navigation accuracy is necessary for successful flights across the buildings and tall beams. In a city setting, drones are especially useful for human and vehicle traffic mapping. The data provided by drones can be used in collaboration with smart phone applications on traffic analysis. The Halton Regional Police of Canada has successfully integrated the use of drones for surveillance [22]. Real-time view of crimes, accidents and individuals that need assistance are beneficial for city authorities. Quadrotors are often used for load transportation. Study [23] applies both a delivery truck and its quadrotor across a residential neighbourhood. The delivery truck is set to stop at multiple points across the environment. The quadrotor then completes the last leg of the delivery process. Both vehicles collectively reduce the operation time by one third. Still, the algorithm isn't readily applicable towards more complex terrains.

The indoor environment poses an extreme hazard to these aerial vehicles as the percentage of clutter as compared to free space is high. The addition of narrow entry or exit ways such as windows and doors are similarly challenging. As shown in Figure 2.8, the Parrot AR Drone 2.0 allows the integration of personal systems such as tablets or smart phones for the navigation for the quadrotor. It is built on the idea that anyone can fly a drone given its simplicity. The drone can be used for indoor flight by attaching its indoor hull [24]. The probability of GPS connectivity indoors may not be guaranteed but the wireless connection that exists in most homes allow for instant navigation. It

also enables high definition video streaming between a tablet and its drone. The wireless connection does have the disadvantage of limiting the flight range to a short distance. AD* search algorithm is applied in [25] to design short term waypoints for a quadrotor that resides within an indoor space. The issue with defining waypoints for a short distance of 20cm is there is a risk of motion towards an obstacle filled region. This can cause the agent to turn around constantly. Similarly, [26] designs a MILP for the trajectory planning across an indoor environment for both the agent and their load. Two subsystems are defined where the first defines the agent with a load whereas the second holds the agent alone. Kinodynamic Rapidly Exploring Random Trees* (RRT) is applied by [27] towards path planning for a quadrotor flying across two windows. In this paper, authors combine both the UAV's control and dynamics along with trajectory planning. Information Rich RRT is applied by [28] to maximize information gathering capabilities of the quadrotor as it flies within a cluttered environment. The planner is also tested within a small sized indoor space. These indoor based path planners would find it tough to plan trajectories across mountainous terrains that have gradual peaks everywhere. Possible future collisions, increased complexity and run time can occur due to the short-term planning that is implemented in these studies.

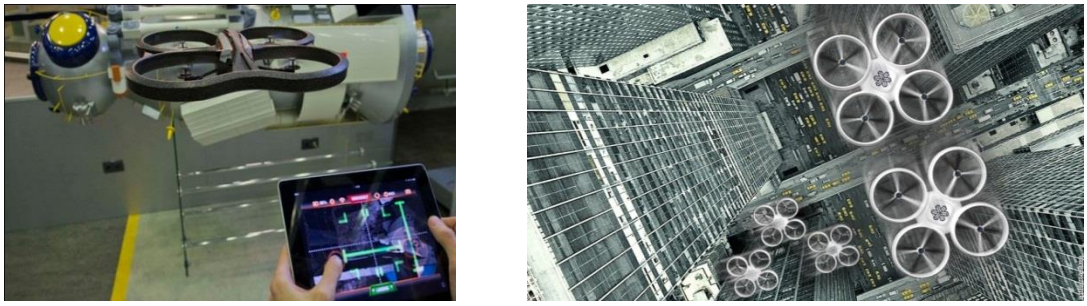


Fig. 2.8. Quadrotor flight through tablet control across indoor and cityscapes [29-30]

Our research applies a path planning algorithm that adapts to many types of environments. The planner can adapt to test spaces that may require low or high-resolution mapping. It is also able to overcome any physical challenges such as sharp bends, narrow passages, high amounts of clutter and gradual terrain peaks. Thus, the end user is given the flexibility of applying the designed algorithm within any environment as opposed to just one.

2.2 TRAJECTORY PLANNING FOR QUADROTOR UAVS

Path planning is typically used to solve the problem of navigating an unmanned robot or UAV from an initial point to a desired destination. In most cases, there are several limitations and constraints that must be adhered to when designing a feasible path. A trajectory planner is applied to provide the end user with an optimal path that satisfies the defined environmental, dynamic and mission constraints.

Firstly, different environments can restrict the motion of a robot in a variety of ways. As described in the previous section, there has been a lot of progress in the variety of environments that the quadrotor UAV can move across. Urban environments have narrow passages and high amounts of clutter across its space. Mountainous terrains or forestations have spaces that are made up of dense flora and fauna. It also has extreme low and high peaks. Trajectory planners must be capable of mapping these environments into either an obstacle region or free space with an acceptable level of accuracy. Secondly, it is also important to consider the dynamic constraints that define each robot. Quadrotors can vary in size, shape, speed and motion. The dynamic model of the aerial vehicle must be based on the physical structure of the quadrotor that is chosen for the path planning experiment. Lastly, the type of mission that requires the agent to fly from its start to goal node can introduce additional limitations. These constraints come in the form of cost functions or objectives that must be considered during a mission.

As previously described, multi-agent quadrotors are often used for a variety of missions. In this study, the missions that are implemented involve multiple agents. The usage of a team of agents can be advantageous but the expansion of an individual platform towards a multi-agent system is a complex process. The trajectory planner must be capable of executing the task that is assigned to each agent simultaneously. These missions can be generalised into two categories which are independent or coupled multi-agent flights. In the case of independent flight, the agents have individually designed trajectories that are free from coupling with the other agents. Here, each agent is provided with its own initial and desired goal node. Independent missions as shown in Figure 2.16 are defined within this study as MA-Spread. Common applications that require the agents to spread across its test environment are such as target tracking, real-time surveillance, rescue missions, wilderness inspection and urban space mapping.

On the other hand, some missions require the trajectories of all agents to be coupled to one another. Cooperative flight is performed through the collective design of trajectories for all agents simultaneously. Here, the path that is designed for one agent is dependent on its neighbouring agent. In this case, the relative distance between all agents is constantly monitored. The distance between any two neighbouring agents is defined based on the current desired formation shape. Thus, the level of coupling of one agent within a formation to another agent is high. Coupled missions as shown in Figure 2.17 are defined as MA-Formation within this research. Flights in formation are used to perform missions such as payload transportation, providing different camera angles, collection of different sensory data and wildlife management.

This section aims to define the different trajectory planners that are often applied towards the multi-agent spread and formation flights. Popular path planners such as sampling, grid and roadmap based planners are discussed in section 2.2.1. These planners can generate individual paths for agents that are flying independently. These

paths can be directly applied within the MA-SPREAD application. On the other hand, these independent paths are used as reference trajectories for the MA-FORMATION application. Thus, section 2.2.2 presents formation planning algorithms that design paths for agents within a formation structure. Trajectory planners for flight in formation often apply a leader-follower system, virtual structure or artificial potential function.

2.2.1 MA-SPREAD PATH PLANNING ALGORITHMS

The process of designing optimal paths for a UAV can be complex. As previously defined, the designer must consider the many constraints that exist within a robotic system and its environment. Expanding this system into a multi-agent system introduces many new challenges for the designer. Therefore, the designer is required to consider issues such as the number of agents, collision avoidance, communication topologies, sensory data fusion and completion of tasks.

Trajectory planning for a multi-agent quadrotor system can be broken down into two stages. The first stage involves the mapping of the test environment. The mapping process can be done through three different well-established approaches. The designer of a path planner can choose to apply any one of them: grid based mapping, sampling based mapping or polyhedral roadmaps. The mapping process creates a geometric model of the world where both the obstacles and free space are well defined. It also produces a weighted graph that mathematically defines the free space. This graph defines the connections between the edges and vertices that are spread across the test environment. The second stage of path planning extracts the best path from the collection of collision free edges. A popular form of grid path extraction for multi-agent robots is through the shortest path Dijkstra's algorithm [31]. Here, collision free nodes are placed within a queue and are defined by a weight. The weight of each node is dependent on the collective cost of the nodes that begin from the start node. The value of each node is constantly revaluated as the planner attempts to define the best path. The best path has vertices with the lowest collective cost. An expansion of the Dijkstra's algorithm is A* and its own extension, D*. In these variations, the objectives of the mission can be used to further determine the most optimal path.

In grid based planning algorithms, the test environment is broken down into grids. The possible horizontal movements of the UAV are shown in Figure 2.9. In [32] a low-resolution grid map is generated for path planning across a three-dimensional test space. Then, initial path planning is performed using A* algorithm. The authors conclude that the paths that are generated through the course grid map are discontinuous at each segment. This is because at the end of each grid segment, the agents are required to perform aggressive manoeuvres to progress from node to node. Grid mapping can be difficult to implement because the size of each grid is defined by the resolution completeness that is required by the end user. Smaller sized grid cubes will increase the processing time of the algorithm. It also has the advantage of producing a more optimal path. Some studies have opted to implement adaptive sized grids to minimize

processing time whilst maintaining a high resolution. The size of the grids also affects the number of nodes within the environment. In this case, the authors have chosen to maintain larger sized grids whilst implementing a higher-level path smoothening algorithm to minimize jerk cost.

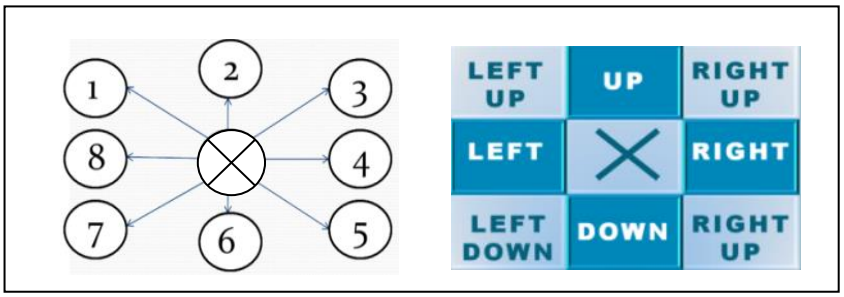


Fig. 2.9. Possible horizontal movements of a UAV across neighbouring grid blocks.

The application of A* algorithm after grid based mapping is advantageous. The combination of the mapping and path search algorithm has simplistic steps that don't require a large amount of data processing or storage. The disadvantage of this simplicity is it can cause evaluations of redundant nodes. Study [33] performs adaptive path planning for three quadrotors that are flying across an indoor environment. Here, a combination of grid mapping and closed loop RRT (CL-RRT) is applied. The algorithm calculates a cost values for each grid block. This cost map is used to create bias in the CL-RRT's sampling process. This process can be challenging to replicate because grid based path planners are effective when the environment is well mapped. The shortest paths can only be discovered if the obstacles are clearly defined during the mapping process.

Another algorithm that is frequently applied towards path planning for quadrotors is Virtual Potential Function (VPF). This algorithm is also known as Artificial Potential Function (APF). VPF performs node to node transitions based on the summation of the attractive and repulsive forces of each grid cube. The attractive field is proportional to the distance between the goal and current flight coordinates of the agent. The repulsive field is inversely proportional to the distance between neighbouring agents and obstacles within the environment. The repulsive field can be designed like a penalty cost within an objective function. A high value can be assigned to the repulsive field when there are obstacles within close range. Real-time flocking of multi-agent quadrotors is presented in [34]. Here, the distances between four quadrotors is maintained through a smooth collective potential function. The agents take more than a minute to arrive at their desired agent-to-agent distances even though the test environment has no obstacles. The most challenging part of VPF is determining the potential value for all agents. This process will require a large amount of processing time. The path planning algorithm will be especially slow within high dimensional test environments or when there are many obstacles.

Study [35] shows the progression of three quadrotors across a forestation. Figure 2.10 shows the potential field for the three quadrotors as they manoeuvre across the forest. The potential field for the goal coordinates of each agent is shown in blue tones. On the other hand, the undesired locations such as the start nodes are highlighted in red. Possible agent-to-agent collisions are also avoided by creating a repulsive field around neighbouring agents. Thus, the advantage of the algorithm is that it can be easily applied towards dynamic obstacles. The algorithm also finds it tough to avoid local minima due to its dependence on local information. It is possible for an agent to be trapped within a local minimum. This can be disadvantageous for path planners that are looking for globally optimum solutions. Thus, designers that wish to apply VPF must also implement an additional local minimum recovery system.

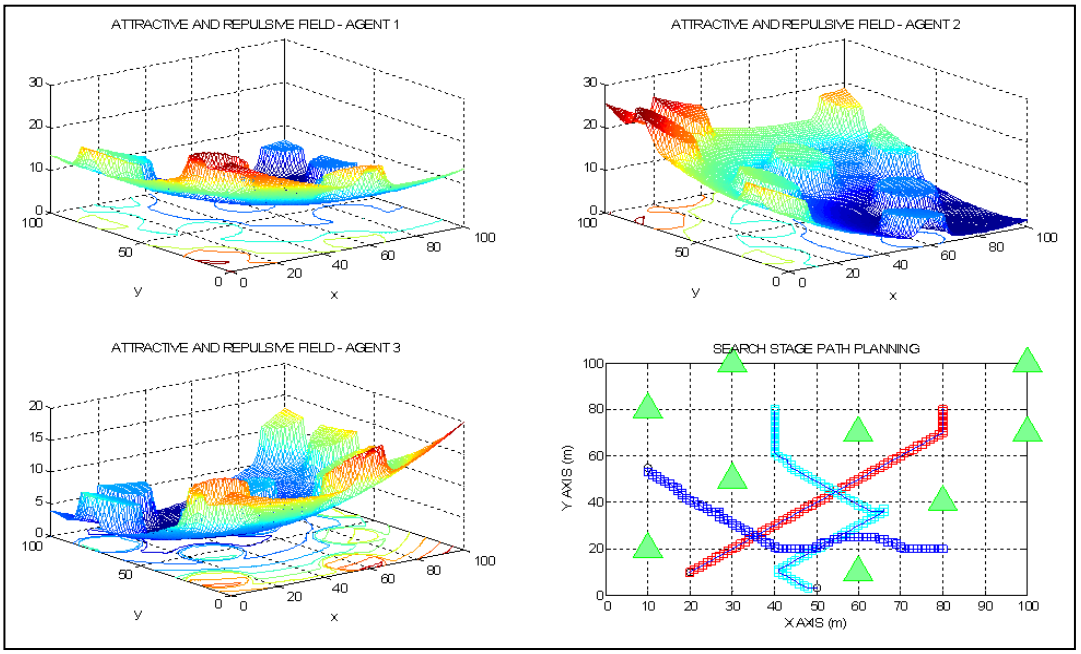


Fig. 2.10. Path planning through Artificial Potential Function for spread flight [35].

Combinatorial trajectory planners are also used to navigate UAVs across various test spaces. This path planner creates roadmaps by making collision free connections between the boundaries of each obstacle. Roadmaps can be generated through several methods. Combinatorial path planners can choose to apply vertical cell decomposition, visibility graph, shortest-path roadmap or Voronoi diagram. The advantage of applying combinatorial methods is its level of completeness. It can find a feasible path or report that a path doesn't exist. The Stanford Testbed of Autonomous Rotorcraft for Multi-Agent Control (STARMAC) compares the usage of VPF and visibility graph towards designing paths for a fleet of quadrotors [36]. Their work concludes that both methods are simplistic and can generate paths quickly. The path that is generated by the visibility graph is shorter but travels closely along all obstacles. On the other hand, VPF generates a path that is longer but has better obstacle clearance. Both algorithms have the disadvantage of designing paths that are dynamically infeasible when used independently. The GRASP lab tests three variations of controllers for multi-robot deployment [37]. The study makes a comparison between a Voronoi-based coverage

control task, probabilistic minimum variance task, and a task using VPF. Their research concludes that discrete mapping functions such as the Voronoi diagram provides less robustness to errors. Continuous mapping functions can produce paths that are easier to track and produce less error.

Study [38] applies a Voronoi-based coverage control for cooperative multi-quadrotor pursuit of an evader. Here, the path nodes of each quadrotor are designed based on a Voronoi diagram that defines the no-fly region. The designed paths move the quadrotors around the no-fly region whilst successfully trapping the evader within it. In this case, the test space is highly simplistic with two obstacles. The disadvantage of applying combinatorial methods is that they are exact algorithms where the boundaries of the obstacles must be well defined. This algorithm is only able to produce an optimal path if there are minimal approximations in determining the corners of each obstacle. If the obstacles aren't well defined, the path nodes may avoid narrow passages and tight spaces. It is also highly challenging to create roadmaps within three dimensional environments. These studies have only tested their algorithms within two-dimensional test environments.

Sampling based algorithms such as probabilistic roadmap (PRM), Rapidly-exploring Random Trees (RRT) and its variations have been used for planning the paths of UAVs across different terrains. Random sampling based algorithms are preferred due to its simplicity and the ability to promote coverage completeness with speed. The key to the sampling based algorithm is its randomized distribution of sample points across the environment. This pushes the focus of the path planner towards mapping the environment as fast as possible. No limitations that are placed on the number of iterations that is required to produce a feasible path. This allows for high levels of flexibility in terms of complexity and running time. It is the end user that decides the level of optimality that is required. The larger the amount of sample points, the more refined the resolution of obstacles and its edges within the space.

PRM generates a roadmap by creating collision free connections between the random sampling points. Study [39] generates trajectories for multi-agent quadrotors through SAFETY-PRM. This algorithm maps environments that have inaccuracies and uncertainties due to sensory error. Their work shows that the SAFETY-PRM algorithm is capable of mapping collision free paths despite mapping errors. Results show that the algorithm requires many samples to fully map the distorted environment. Like other roadmap based algorithms, PRM generates many redundant connections between the sample points. [40] also applies PRM towards trajectory generation for a quadrotor within a three-dimensional environment. The test environments such as indoor spaces and caves are used within their work. This study combines PRMs with Nonlinear Programming (NLP) to generate dynamically feasible paths. Results show that the PRM-NLP hybrid algorithm can generate optimal paths across these complex environments in a short amount of time. The authors conclude that sampling based algorithms such as PRM can be further sped up through parallel processing.

After analysing the trade-offs between the different path planners, this research performs initial path planning through another sampling based planner which is RRT. This algorithm has many advantages such as:

- *The algorithm is probabilistically and resolution complete.* The basic RRT algorithm is often applied due to its ability to fully explore a space within a short amount of time. With a multi-agent system, multiple trees can be generated to promote better coverage and speed up the exploration process.

- *The concept of RRT is direct and noncomplex in nature.* Unlike prior path planning algorithms, RRT doesn't require explicit construction of the obstacles within its test space. It performs free space mapping through random sampling of the environment. These samples connect to form collision free path subsections and are easily stored and extracted from a mutual database.

- *Creates minimal buffer range between path nodes and obstacles.* This process easily maps narrow passages and sharp corners without a large buffer space between the obstacles and the agents. Thus, shortest paths are easily obtained.

- *Minimal redundant nodes.* The sample points are well spread across the test environment. There are no limitations that are placed on the distance between two samples unlike path planners that use grid blocks. This reduces the number of redundant nodes across a path.

- *Easy implementation of heuristic function*. The end user can optimize the sampling process by introducing a heuristic function. The sample nodes can be biased towards the goal node, obstacle free zones or shortest distance.

The RRT planer is inspired by the way tree branches in nature grow within their environments. A tree trunk is rooted at a location and has many branches attached to it. These branches continuously build upon each other. This creates a parent and child relationship between the main branch and its smaller branches. In time, the branches are well spread across the forest. Trees that are older in age typically have more dense branches and leaves. As shown in Figure 2.11, RRT applies the same concept of parent and child branches that are used to build full trees. The creation of the parent-child connection is less complex than building a roadmap. Firstly, each tree is rooted at the UAVs predefined start and the goal nodes. Sample points are then placed across the test space and are connected to the nearest collision free tree branch. The planner works by incrementally building these branches as the iterations progress. These branches quickly spread across the test environment and become denser with time. Some works implement multiple trees that are rooted at different locations. The branches of these trees explore different areas of the test space and advance towards each other using a

greedy heuristic. This allows the end user to implement high resolution mapping when necessary.

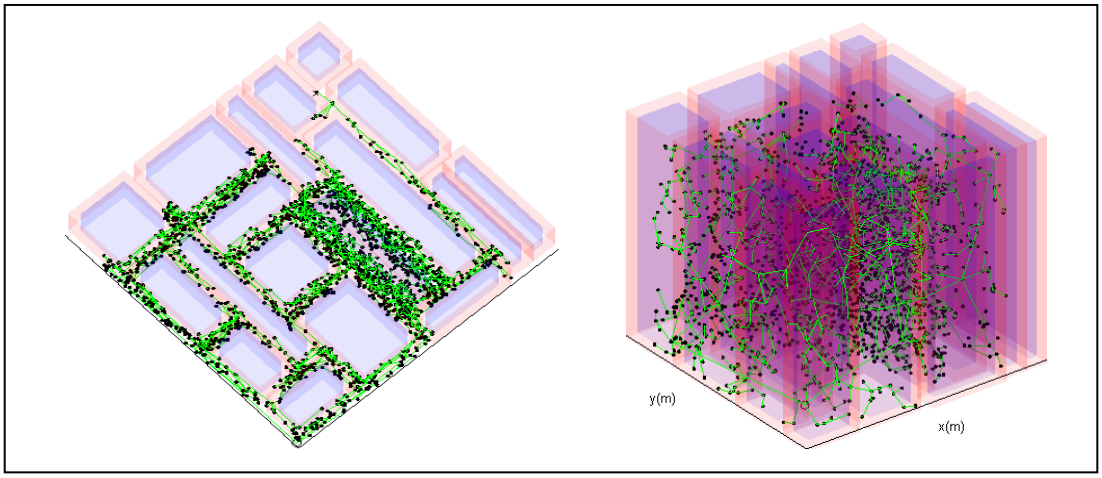


Fig. 2.11. Single RRT tree across cityscape environment

Like other sampling-based planners, many variations of the RRT algorithm apply adaptive sampling methods. This encourages its usage within complex environments. The end user has the option of placing a larger number of samples across narrow passages or highly cluttered areas. The user is also able to implement path optimization during the sampling process. In this case, the sample points are connected to the branch with the lowest cost value. Other strengths of the RRT algorithm is its speed, intuitive progression of nodes, near optimality and free space probabilistic completeness. Thus, the application of randomly-exploring random trees as a tool for path planning is advantageous. The final path can be filled with aggressive bends due to the structure of the tree branches. The addition of trajectory smoothening mechanism can reduce the impact of extreme bends within a path. Smooth splines that create minimal jerk trajectories can be applied after the path planning process. Splines are easy to implement in comparison to other polynomial equations because can generate a trajectory for an entire path simultaneously through a simple recursive function.

Numerous studies apply sampling-based algorithms such as RRT because it can promote complete coverage with speed. Research by [41] focuses on applying closed-loop RRT (CL-RRT) with an autonomous ground vehicle. This real-time application of RRT with a dynamically unstable vehicle shows that it can generate path nodes with speed. Many authors have chosen to explore modified versions of RRT. In study [42], RRT* is used for initial path generation. These initial paths are then transformed into trajectories through polynomial splines. The algorithm can perform well with one quadrotor. Expanding the RRT* algorithm towards a multi-agent system will require a higher sampling rate and processing time. In a bid to reduce complexity, the algorithm doesn't fully consider the dynamic capabilities of the quadrotor. The authors of [43] apply Information-Rich RRT (IRRT) to maximize information gain whilst minimizing travel distance. The IRRT can plan paths for both cooperative and non-cooperative

multi-agent quadrotors. In both cases, the agents are planning paths independently whilst broadcasting path information. Thus, the agents may perform sampling across similar areas. This can be disadvantageous because the agents are collecting redundant information. The algorithm can benefit from additional optimization.

Next, Desaraju and How [44] design a decentralized path planner for multi-agent teams in complex environments using RRT (DMA-RRT). Their algorithm allows an agent to modify the paths planned by other agents. This option allows each agent to make improvements that reduce collective costs of a group. Thus, the global performance of a team is preferred over individual excellence. A token system based on potential improvement is used to decide which agent plans their path next. To implement the DMA-RRT algorithm, two subsections should work together. Here, the first section is the individual component that implements agent path planning. The second section oversees the interaction component processes between agents. This section ensures that all the information is shared between all agents. Results show that a significant reduction in run time isn't seen with a small number of agents. Study by [45] also applies a modified RRT towards multi-UAV planning in obstacle rich environments. The anytime RRT algorithm must be able to avoid static, pop-up and dynamic obstacles within the free space. The RRT algorithm is used to carry out a new search from the current location of the UAV if the agent encounters anything dangerous. Path replanning is also performed if an agent deviates from the designed path. Results show that the algorithm can avoid high amounts of clutter and finding a path within a short time.

Biderectional or multi directional RRTs (mRRT) are more current variations of the basic RRT algorithm. Recent times show progress in the multi-tree RRT [46] field. Here, more than one tree is constructed in series or parallel. This allows the trees to be rooted at different locations across the environment. The idea of multiple trees is advantageous because there is more coverage within a shorter amount of time. The mRRT can also expand across places of interest such as local minima or dead ends. The application of mRRT towards multi-agent UAV system is best with the merging or creating connections between trees. This allows the sharing of information and cooperative path planning. Study [47] applies a multi-directional Rapidly Exploring Random Graph (mRRG) where the tree is expanded towards multiple directions. This method produces faster space coverage and pathway generation since there are more branches. The process of mapping can be further improved by the using multiple trees [48]. Research by [49] uses transition-based RRT where a few trees are built from different root locations. The algorithm is only implemented for the extraction of a singular path.

The next variation of the RRT algorithm is the hybridization of multiple extracted trajectories. Work by [50] applies RRT for the generation and merging of pathways between protein conformations. Here, 100 paths are extracted through the basic RRT algorithm. The nodes within these paths undergo clustering, hybridization

and ranking. The aim of the hybridization process is to create one superior path. The clustering and merging of these paths are based on the values of the objective function that is used. Whilst this process is effective for one path, implementing it for a large collection of paths will be more complex. This is due to the high number of comparisons that are necessary between each path's nodes.

Our study applies both the multi-tree and hybridization process. Here, the CL-RRT includes the limitations and constraints that are present within the quadrotors. Processing time is reduced by sharing sampling points and mapping information between all agents. These initial paths are then applied towards the creation and optimization of new paths.

2.2.2 MA-FORMATION TRAJECTORY PLANNER

The prior subsection defined the path planning algorithms that are used to generate paths for each agent independently. In most cases, similar algorithms are applied to generate the reference path for agents in formation flight. These reference paths are transformed into trajectories for each agent within a formation. The variety of methods that are used by researchers to design and maintain formation flight is discussed within this subsection. Many works have taken inspiration from Reynolds' three rules of flocking behaviour [51] when designing a path planner for formation flight:

1. Maintaining cohesion: The agents constantly steer towards the average position of its neighbouring agents.

2. Creating separation: Agents are to fly away or maintain a safe distance from its neighbouring agents.

3. Preserve alignment: The multi-agent system heads in the same direction.

Planning for multi-agent flight in formation requires a three-stage system. Firstly, the design of the reference trajectory across the test environment is performed. Next, a formation planner is utilized to design the formation shapes across the trajectory. Lastly, the trajectory for each agent is generated based on both the reference path and formation shape. As shown in Figure 2.12, flight in formation means that the trajectory of each agent has a high level of coupling with its neighbouring agents. The most popular method for formation planning is the leader-follower system. In this case, the following agents track the movements of the leading agent. Here, the agents are required to maintain a set distance from a reference trajectory. Formation flights are often successful through the usage of consensus algorithm. In this case, the agents communicate with each other and cooperatively maintain their desired formation shape. VPF is also often applied for formation flight. The structure of the algorithm remains the same as typical MA-SPREAD path planners. The only difference here is that the attraction force is proportional to the distance between the following agent and the

leading agent. The attraction force can also be proportional to a formation reference path. Another common way of generating the trajectories for the agents in formation is through a virtual structure. Here, the independent trajectory for each agent is based on the desired formation shape as shown in Figure 2.12. Lastly, behaviour based systems that mimic the motions of animals in nature such as birds and herds are also used for formation planning.

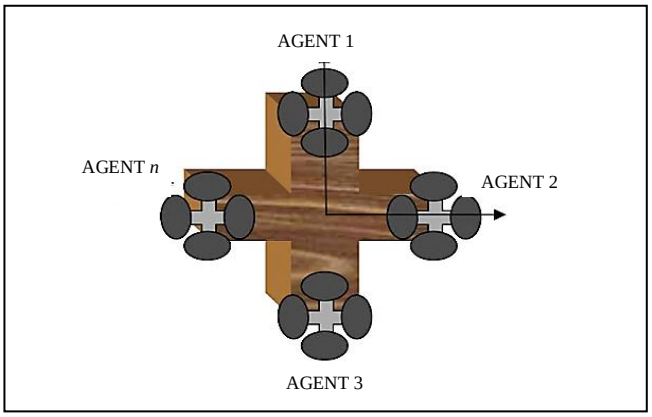


Fig. 2.12. Rigid formation structure based on the desired formation shape.

As previously described, the leader-follower formation system is extremely popular in current times. The downside to this system is its full dependence on the leading agent. In centralized systems, the collapse of the formation is possible in the event of a faulty leading agent. Many researchers have attempted to reduce the risks that come with a leader-follower system whilst maintaining its simplicity. Work by [52] produces paths for a leader-follower multi-agent quadrotor system. The following agent is required to maintain a fixed distance and angular deviation from the leading agent. Their research uses fuzzy logic and GA to preserve the trajectories of the following agents in case connection is lost with the leading agent. Here, an estimation of the leader's path must be obtained. Even though their work tries to minimize the damage that can occur from a centralized system, there can be a large error when estimating the trajectory of the leader. It would be tough to minimize the estimation error especially when sudden and aggressive manoeuvrings are performed by the leading quadrotor. Another group of researchers have published multiple papers on the leader-follower system for quadrotors [53-55]. Their work tests the stability of the system by arbitrarily switching or adding weights to the formation topology. This process allows the formation team to have more than one leader or have its leader be interchangeable. Results show that the agents still achieve consensus with a switched topology. More stability can be introduced by reducing the impact of a faulty leader. In this case, any healthy agent can be switched to the leading agent. Still, there is dependence on the leader's reference signal to plan the paths for the following agents.

A modified leader-follower (MLF) system is applied towards transporting a suspended payload [56]. The authors utilize four quadrotors with a suspended load in a ring topology. Unlike previously discussed works, the MLF system creates dependence

between both the leader and its follower. The movements of the leading agent are affected by the feedback that is received by its following agents. Results show that both the following and leading agents can cooperatively change velocities during a fault and maintain formation. Another study that creates dependence between the leader with its followers is [57]. This work stabilizes the movement of the leader through a reference frame that is formed by virtual followers. These virtual vehicles are placed at a predefined distance vector. Here, the control law pushes the leading agent to maintain the desired distance from its follower as opposed to the other way around. The same team designs the trajectory of a following agent through the positional derivatives of the leading agent [58]. This study shows the ability of the real follower in tracking the motion of the virtual follower. The virtual follower moves like a trailer. Thus, the trajectory for the virtual trailer contains the positional derivatives of the leading agent. This system works like a fully decoupled path planner once the agents are flying/online. The following agents independently track the path of their virtual trailers. These works are innovative and are challenging the high level of coupling that exists within the leader-follower system. Still, the agents are dependent on each other to change and maintain a formation shape. This can be challenging in environments that have a large amount of clutter. The error of one agent is always propagated to the others within its team.

Artificial potential function (APF) is often applied for the generation of formation trajectories. Figure 2.13 shows the APF forces that are applied whilst defining the next node within the formation path. The blue coloured areas define the attractive force whereas the red areas define the repulsive force. The cumulative force for the leading agents is similar to the independent MA-SPREAD scenario as shown in Figure 2.10. With the following agents, the attractive forces are used to keep agents close to the leading agents and in formation. The repulsive forces are used to avoid collisions with neighbouring agents. This can be seen in Figure 2.13 where the blue area is close to the current location of the leading agent. Whereas, the red areas are defined as the parts of the terrain that are far away from the leading agent. Applying APF for multi-agent flight can be highly advantageous in test spaces that have many dynamic obstacles. The simplicity of the APF equations allows the agent to quickly detect a moving obstacle and plan a new direction. Defining both forces constantly is important for successful formation maintenance. Due to this, APF requires constant sampling of the environment which leads to a high processing time. Thus, APF can be beneficial for short distance formation planning but will increase in complexity across large or high dimensional spaces. Another disadvantage of APF is the possibility of an agent being trapped in local minima. This situation happens when the summation of the attractive and repulsive force is zero. In this case, the agent is trapped in this location. Thus, the path planner is incapable of exiting the local minima and moving towards the goal node without additional help.

Research by [59] shows the basic application of APF for formation planning with a team of three quadrotors. Their potential function is a sum of three forces. The

first force defines the position error of the agents based on the desired formation shape. Next, collision avoidance is taken into consideration by determining the distances

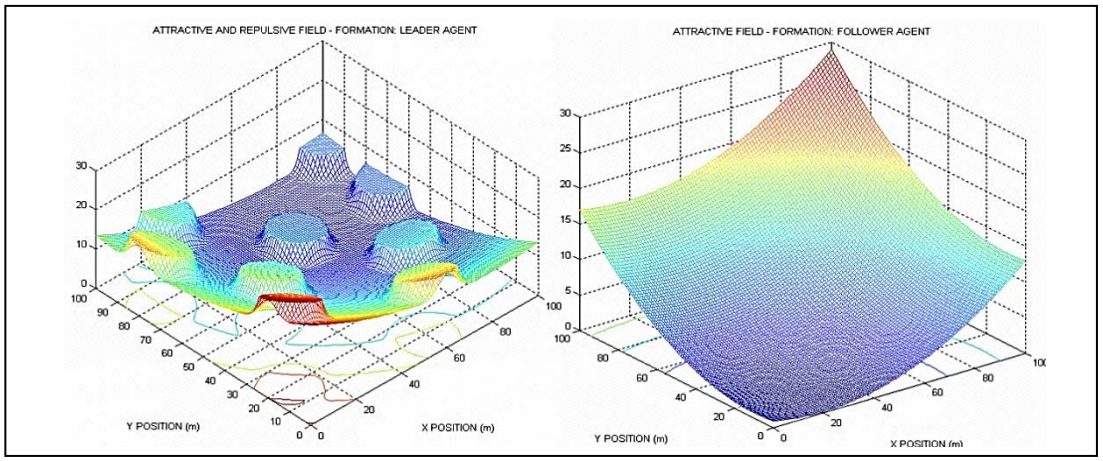


Fig. 2.13. Path planning through Artificial Potential Function for formation flight [35].

between neighbouring agents. Lastly, the final force pushes the agents to track the trajectory of a virtual leader. Many modifications have been introduced to simplify the full potential function. Work by [60] uses APF to design the trajectories of six quadrotors that transition between three different formation shapes. Initially, the agents form a two-dimensional star shape. They then move to create a rectangular and triangular shape. Here, only close-range obstacles are considered within the path planning algorithm. This process reduces the processing time that is required to determine the repulsive forces.

The same team further simplifies the repulsive force equation by eliminating the need for agent-to-agent collision avoidance in [61]. The authors perform strategic goal node assignment for a group of 10 quadrotors. Each agent is assigned a goal node whilst transitioning from one formation shape to another. The path planner designs a trajectory that allows each quadrotor to fly from to their goal node without colliding with any of their neighbouring agents. There is no need to constantly define the agent-to-agent repulsive force if the goal node of each agent is well assigned. Though both these studies have attempted to simplify the APF equations, it is performed in a 2D environment with no more than two small circular obstacles. These improvements may be negligible in highly cluttered 3D environments. Researchers have also attempted to reduce the effects of being trapped within local minima. Study [62] implements a wall-following system to guide an agent out of the local minima region. It is a simple system that tells the agents to fly close to the obstacle's boundary and escape local minima. There are two disadvantages to this system. The boundaries of each obstacle must be accurately defined so that there are no collisions whilst wall-following. The agents will also have to travel longer distances. A path planner that isn't affected by local minima will produce shorter paths.

Consensus algorithm (CA) is constantly applied for multi-agent formation planning. This provides more robustness as compared to previously defined centralized leader-follower configuration generated with APF. Consensus produces a decentralized structure where the death of an agent or the loss of communication link is not detrimental to global coordination. Here, graph theory is applied. A graph is modelled as a collection of vertices and edges. An edge (communication link) is a connection between two vertices (agents). Next, an adjacency matrix is applied to imply the options for networking from agent to agent. A team of multi agents that exchange data are typically modelled by directed or undirected graphs. Directed graphs are graphs which have a direction associated with each edge. Undirected graphs promote two-way communications between the vertices and its edges. Multi-agent systems that are well connected can reach consensus quicker than partially connected systems as shown in Figure 2.14. The communication topology is often assumed to be time varying due to vehicular motion or communication dropouts. These can be caused by propagation loss, diffraction and noise disturbances.

CA minimizes processing speed by focusing on agents that are within each other's communication range. The downfall of communication links between agents can be overcome as well. This is due to the inclusion of multi-agent network topology within the formation control structure. Thus, consensus algorithm creates fully distributed and fault-tolerant formation architectures. As with centralized structures, fully distributed systems can have some disadvantages. An agent can be left out of the communication chain if connection is lost with its neighbouring agent. A distributed system also requires the full collaboration of all agents to form different formation shapes. This process can require a lot more time than a centralized system since the cooperation of all agents are important.

There are a few current works that have applied CA towards maintaining the formation shapes of multi-agent quadrotors. Study [63] uses a sliding mode control with CA for path planning and tracking. Three quadrotors are required to fly from their initial position and form a triangular formation structure. Their work shows that many types of communication topology can be used to achieve consensus. These agents perform formation flight with a fixed, directed spanning tree or undirected network graph. In all cases, the quadrotors could create and maintain its formation shape in less than 5 seconds. Still, the study assumes that all agents know their desired trajectories and have the same trajectory within an obstacle free space. Similarly, study [64] applies the algorithm with three quadrotors that aim to maintain a rotating triangular formation structure. The multi-agents have a directed communication topology. As previously defined, this can be dangerous if an agent loses communication with its neighbour.

Work by [65] combines a leader-follower system with CA towards formation planning for a group of quadrotors. The agents have indirect contact with the leading agent as well as its team mates. It removes the need for all agents to be directly connected to their leader. Thus, the positional information can be passed on despite

losing contact with the leading agent. The same team aims to include obstacle avoidance into their algorithm by implementing APF as well [66]. Here, only agent-to-agent collision avoidance is tested. Each agent has a cylindrical shaped safety zone around it. These safety zones define the value for the APF's repulsive force. In these studies, very simplistic conditions are applied. They don't really challenge the capabilities of the CA. The running time of the planner will be much longer when the agents should transition between multiple shapes. Predicting the flight time that is required by each agent between formation shapes can be challenging with CA. Another challenge with CA is its ability to plan paths that avoid obstacles. It must be used in collaboration with another algorithm to effectively avoid all obstacles within the test environments.

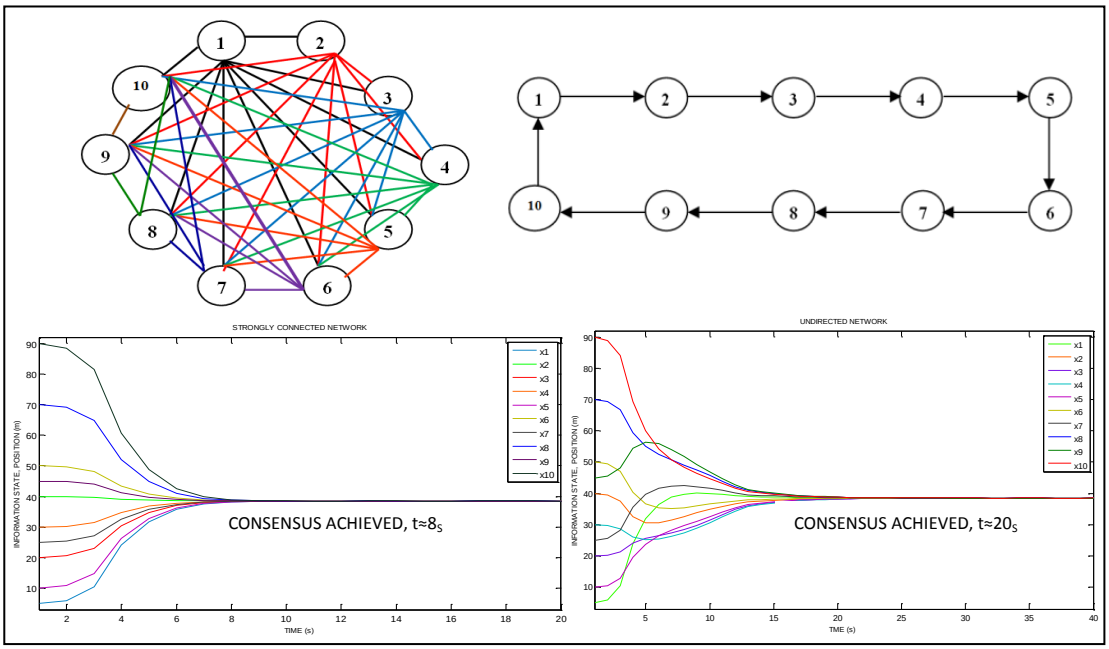


Fig. 2.14. Speed of convergence for strongly connected and undirected network.

Upon weighing the pros and cons of each method applied for formation planning, this research applies a virtual structure (VS) or virtual rigid body for fast and stable formation flight. There are many benefits to using a virtual rigid body:

- *This system isn't fully centralized or decentralized*. There is less coupling between agents. Each agent is less affected by the death and faults of their neighbours. Here, both the VS and the reference trajectory are used to create independent paths for each agent. Figure 2.15 shows a circular VS that is used to generate the trajectories for 10 agents. Initially, it is centralized in the sense that a reference trajectory is necessary. Once in flight, the agents are flying independently and are fully decentralized.

- *The application of a closed loop multi-agent UAV prediction system.* The independent trajectories can be easily applied for an estimation system that

predicts the movement of each agent. Thus, a group of heterogeneous quadrotors with different physical capabilities can be flown simultaneously. The trajectories will be dynamically feasible for each individual agent based on their physical capabilities.

- *It also removes the dangers of lost communication links between agents.* The agents can maintain their formation structure since they are tracking independent trajectories. These trajectories are not dependent on their neighbouring agents. Thus, only basic collision avoidance is required between two agents whilst they are flying in formation.

- *Minimal positional and rotational error propagation*. Here, each agent is supplied with an independent control system. The lack of coupling between each agent during flight reduces the propagation of any agent's error across the team.

- *Easy execution of many complex formation shapes*. The compression or spreading of the multiple agents in formation can be performed by changing the shape of the virtual body. This can be advantageous for environments that have large amounts of clutter or narrow spaces.

- *No need to define the environment or its obstacles.* The virtual body changes shape and size based on the free space contour around each path node. If this is well executed, each agent doesn't require much information regarding its environment. Thus, only dynamic obstacles need to be identified.

There are a few researchers that have applied virtual structures as shown in Figure 2.15 within their formation planner. [67] addresses one of the challenges that can occur when designing VS across a multi-agent quadrotor formation trajectory. In most cases, the shape of the VS determines the positional and rotational derivatives of all the agents in formation. They acknowledge that the agents within a team may not all be capable of achieving and maintaining a designed VS. This happens when the VS progresses faster across the reference trajectory in comparison to the agents. Thus, the agents are incapable of tracking their trajectories. Their work introduces trajectory replanning if an agent is lagging. The replanning process slows down the entire formation so that the lagging agent can keep up. If this isn't possible, the other agents progress without the faulty quadrotor. Another study implements a virtual rigid body with a group of quadrotors [68]. This system decouples the virtual body's movements from the trajectories of each agent. Here, the quadrotors are required to transform into six different formation shapes. Results show that the agents can transit between these shapes quickly. This shows that a formation path planner that is decoupled from its shape planner can minimize processing time. The disadvantage of this system is that the formation paths are not proven to be dynamically feasible. Also, the formation shapes are randomly determined because the test spaces have no obstacles within them.

Formation planning with virtual rigid bodies is ideal for this thesis. The concerns that are raised within prior works are tackled within this research.

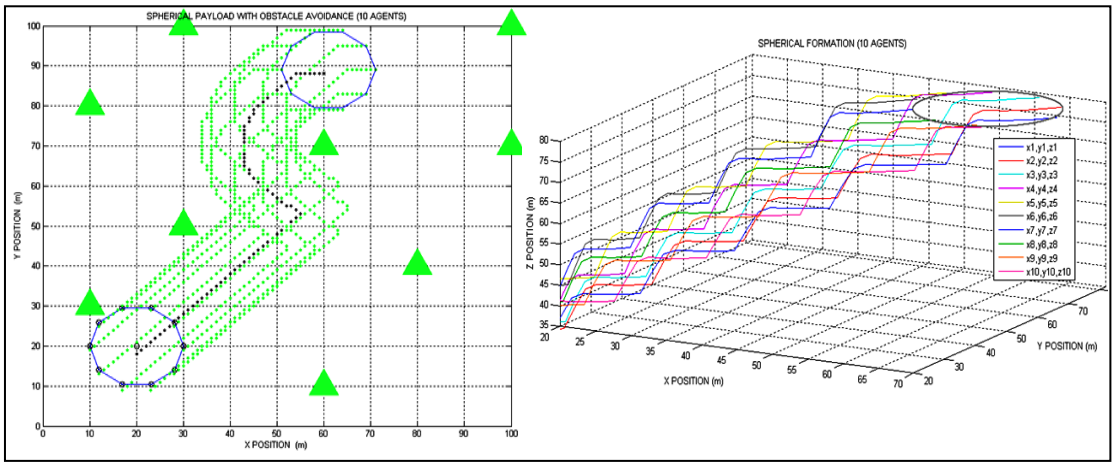


Fig. 2.15. Path planning for 10 agents in formation through a spherical virtual structure.

In this thesis, the formation shapes for eight quadrotor agents are designed based on the obstacle free space contour around the agents. The contour is mapped through the radius of free space at all angles from the formation waypoint. The virtual structures across the formation path are designed based on the size of the free space contour. Next, our formation planner designs both rigid and nonrigid shapes for long distance flights. It is capable of a high number of shape changes that adapt to its environment. No limitations are applied upon the network topology since the agents will be tracking independent trajectories. The quadrotors will maintain a safe range from each other using sensory data. Thus, this system collectively produces a formation planner that is high resolution and adaptive to complex environments.

2.3 MULTI-AGENT QUADROTOR MISSIONS

With any real-life decisions, the pros and cons of all options are weighed before making a choice. Objective functions are defined as mathematical representations of the pros and cons of various options. Thus, different choices produce different objective function values. Ideally, the best choice is one that considers all objectives fairly without extreme sacrifices of one cost function over the optimization of another. In reality, achieving fairness is extremely hard. Similarly, when designing the trajectory for a multi-agent system, various objectives are applied to form a collection of trajectory choices. These choices can be narrowed down based on the user's preferences. The accurate detection of necessary objectives is important as complexity increases with the number of cost functions. In cases where too few objectives are present, the final trajectory will be biased only to those objectives present. This can cause degradation in costs that were not defined. Here, the various types of objective functions that are applied within multi-agent trajectory generation algorithms are discussed.

2.3.1 OBJECTIVES OF SPREAD FLIGHT

In this study, multi-agent spread missions are defined as tasks that require the agents to fly across the terrain independently. An example of spatially spread flight is shown in Figure 2.16. Thus, the path for each agent is designed based on their predefined start and goal nodes. Common spread missions are target tracking, search and rescue, environmental mapping, human-assisted navigation, load lifting and agricultural surveillance. In most cases, the agents are used to collectively explore uncertain areas. Trajectory planners that are designed for quadrotors are often focused on path length and altitude, aggressive manoeuvring, path tracking error, time optimality as well as fuel consumption. It is also important that these planners also consider minimizing possible collisions and network decay. Lastly, efficient environment mapping is encouraged through the exploration of uncertain areas and the reduction of redundant sensory data.

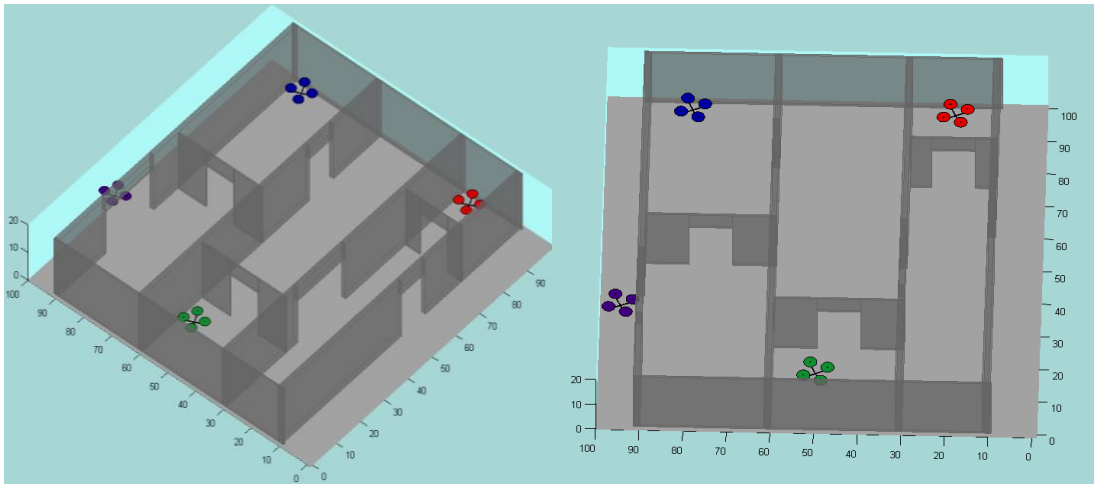


Fig. 2.16. Independent multi-agent quadrotor flight across an indoor space.

The most common requirement for a trajectory planning algorithm is to generate the shortest path. This objective is highly important because it minimizes the fuel consumption thus leading to a longer flight time. The shortest path is influenced by the path's node-to-node distance which also takes into consideration the altitude of the UAV. Target tracking is a popular application for multi-agent quadrotors. In many cases, the agents should maintain visual contact with a few targets simultaneously. This process is made up of two sections which is target searching and tracking. In [69], the path planning algorithm minimizes two objectives which are the travel distance and target pose uncertainty. The aggregated cost function allows the end user to define the importance of finding the shortest path. The level of importance of an objective within a cost function is set through predetermined weights. Another objective that facilitates the extraction of the shortest path is the minimization of goal deviations. This cost penalizes node to node progressions that move further away from the goal node. A swarm of 20 micro quadrotors are flown across a known indoor environment in [70]. This study aims to minimize the goal node cost through optimal goal assignment. Here,

the goal nodes aren't attached to an individual agent. Each agent is free to progress towards any goal node to minimize the total cost function.

The advantage of using UAVs as opposed to ground robots is its ability to fly across three dimensional spaces. Path planners must consider the altitude of each agent. Quadrotors often fly at higher altitudes because it avoids colliding with obstacles across each terrain. Still, flights at higher heights require a large amount of fuel, thrust and climb. Thus, a multi-agent path planner must be able to determine the best flight altitude for each mission. As previously mentioned, the transportation of loads has become a common application for multi-agent quadrotors [71]. It is important for the agents to maintain flight at certain heights due to the load that is placed beneath them. The load lifted by the agents can crash against the ground if it is flown at a low height. On the other hand, it can cause an increase in fuel consumption if flown at high heights. Faust *et.al.* generate trajectories for a quadrotor with a suspended payload [72]. Path nodes are placed at an appropriate height with low amounts of oscillations. These costs are prioritized to minimize load swinging. Wang *et.al.* aims to design trajectories for quadrotors that are flying across a partially known indoor environment [73]. The agent's mission is to follow the smooth trajectory through a window and drop payload at a designated target location. In this case, the flight height is extremely important since each window creates a border around the agent and its payload. Similarly, the possibility of applying the multi-agent quadrotors for cooperative constructions is explored in [74]. These agents are required to construct structures such as cubic, pyramid, tower and wall using nodes and beams. Here, the altitude of each agent is determined by the location of the building blocks and the height of the structures within the construction site.

Another objective function that offers the end user flexibility is the smoothness of a path. The application of multi-agent quadrotors is advantageous because it is capable of aggressive manoeuvring at high speeds across various terrains. The agents must be capable of flying across sharp bends with minimal vibrations whilst transmitting real-time imagery. Many studies prefer to choose smoother paths that allow the quadrotors to transition from node-to-node smoothly. Study [75] attempts to improve the quadrotor's ability to undertake aerobatic manoeuvres with minimal error. The authors apply a control system that can compensate for any altitude error during turns. The control system produces a much flatter trajectory by predicting that a sudden increase in thrust will occur during sharp bends. Polynomial trajectory planning for aggressive quadrotor flight is presented in [76]. Their work implements a cost function that penalizes the squares of the positional derivatives. Here, the end user can define if the minimization of jerk, snap, crackle or pop value is preferred.

Many studies are focused on obtaining trajectories that encourage minimal path tracking error. Tracking error is dependent on the feasibility of the designed path. The path planner should consider the size, thrust and speed limitations of each agent. A quadrotor simulator for outdoor flight is designed by [77]. Their model implements

both environmental and mechanical factors to estimate and minimize position error. Research by Tomic *et. al.* design a platform for search and rescue missions within indoor or outdoor environments [78]. The system architecture merges data fusion, mission control and path planning subsystems to effectively track paths. Experiments show that the quadrotor can estimate and follow the desired position with minimal error. The quadrotor is often applied across different environments and weathers. In harsh environments, it can be difficult for the quadrotor maintain minimal path tracking error. Guerrero *et. al.* tests a trajectory planning in known wind fields for unmanned quadrotors [79]. In this case, the planner must be capable of designing and tracking trajectories despite windy situations. Here, the path cost function is a combination of constant and varying wind variables. These studies show that it is advantageous for trajectory planners to include path smoothness within its chosen cost functions. The agents may not be capable of tracking these paths in real-life if these constraints aren't considered at the initial stage of path planning.

Another field that is constantly explored by researchers is the creation of time optimal trajectories. The paths are constructed to mimic real-life flights. Here, the physical limitations of each agent are taken into consideration. Study [80] generates time optimal paths for quadrotors that guarantee dynamic feasibility. These trajectories are designed to allow the quadrotor to complete the mission within minimal flight time without exceeding its speed and acceleration limitations. Many studies also aim to prioritise trajectories with minimal fuel consumption. Chamseddine *et. al.* [81] produce works on the planning and replanning of minimal energy trajectories for a quadrotor. Focus is placed on obtaining paths with minimal flight time despite actuator constraints and faults. The path planner is assisted by faults-tolerant control (FTC). Study [82] uses different sized quadrotors for the Sensing Unmanned Autonomous Aerial VEhicles (SUAAVE) project. These agents are used for search and rescue operations. Extracting paths with minimal flight time can be challenging when multiple objectives are considered at the same time. This study prioritizes objectives such as the amount of energy consumption, possible collisions and obstacle avoidance. The amount of data sharing between agents is also considered for trajectory planning. Another study that combines energy management with other objectives is [83]. Here, the path planner aims to conserve and recharge the energy required by a swarm of quadrotors. The batteries are recharged through Ground Recharge Stations (GRS) that are placed across the test environments. Other objectives that are considered are such as the mission status, number of agents and possible faults. Thus, the path planner must be capable of finding paths with minimal fuel consumption without sacrificing the minimization of other objectives.

Multi-agent quadrotors that fly independently can collide with each other if the position of each agent isn't considered. A robust communication network between all agents is important for agent-to-agent collision avoidance. A minimal decay connection allows each agent to estimate the states of other agents with high accuracy. Defining the network topology and possible delays between the agents within a team is

prioritized in study [84]. Proper estimates of each agent's state are only possible with a stable communication link that has minimal data packet loss. This objective is highly dependent on the transmission range of the antennas that are used on the quadrotor agents. The communication between agents is also extremely important for avoiding agent-to-agent collisions. In [85], five quadrotors flew autonomously whilst avoiding agent-to-agent and obstacle collisions. Here, some agents are able to fly across short and direct paths. The remaining agents had to fly across longer paths to avoid colliding with another agent. Thus, finding a balance between obtaining the shortest path whilst avoiding collision becomes challenging. Similarly, [86] applies 50 simulated quadrotors towards creating and displaying a 3D animation of a human in motion. The position error of the agents must be well minimized to avoid collisions whilst properly shaping and visualizing each animation. These studies show that it is very important for designers of a multi-agent system to maintain a good communication link between the agents.

Quadrotors are also capable of efficient information collection through sensory fusion. Multiple agents are typically spread across unknown environments to perform real-time mapping. Here, trajectory planners must encourage their agents to fly across uncertain areas within the terrain. This process reduces the amount of redundant sensory data by prioritising paths that are diverse in direction. In [87], quadrotors are deployed simultaneously to increase situational awareness and track targets within its test environment. Similarly, study [88] applies a multi-agent quadrotor system towards aiding operators in humanitarian demining. The agents are used to provide aerial imagery of harsh environments. There are two objectives that are important in both these studies. The first is to maximize the amount of space exploration and the second is to minimize similar data collection. Work by Soltero *et.al.* prioritizes the maximization of information collected within unknown environments [89]. Their algorithm designs paths for a quadrotor that flies across dynamic environments. These environments rapidly change as the quadrotor flies across them. Results show that the adaptive path planner can map the environment and producing a path within a small number of iterations. Similarly, study by He *et. al* produces trajectories through a Belief Roadmap algorithm within GPS denied environments [90]. The algorithm can properly identify obstacles within the free space through non-uniform sampling despite sensory limitations. The works that are described placed a lot of importance in collecting adequate information for environment mapping, target tracking and path planning.

In real life, many objectives are important to the successful completion of a spread mission. The objectives are typically tailored to an application. Many works choose to exclude some objectives or create a priority system that reduces the importance of certain objectives. More flexibility and knowledge can be obtained when the focus is on the optimization of all objectives. This research chooses to optimize all objectives equally through many-objectives optimization.

2.3.2 OBJECTIVES OF FLIGHTS IN FORMATION

The second application that is explored within this study is flight in formation. An example of formation flight is shown in Figure 2.17. Quadrotor UAVs are capable of aerial flights whilst maintaining precise patterns. Formation flights are crucial for tasks such as payload transportation, security patrols, search and rescue or environment mapping at hazardous sites. Studies on multi-agent formation flight have implemented a variety of objective functions. Like spread missions, paths that have minimal length, altitude, goal deviations and aggressive manoeuvring are prioritized. Time optimal trajectories that have minimal flight time are also preferred by most researchers. A highly important objective for formation flight is the reduction of positional error. The minimization of each agent's positional error will also create a reduction in possible agent-to-agent collisions. The cost functions that are unique to formation flights are based on the designed formation shapes. Here, many works apply objective functions that minimize the number of formation shape changes, scale complexity, maintenance and rise time.

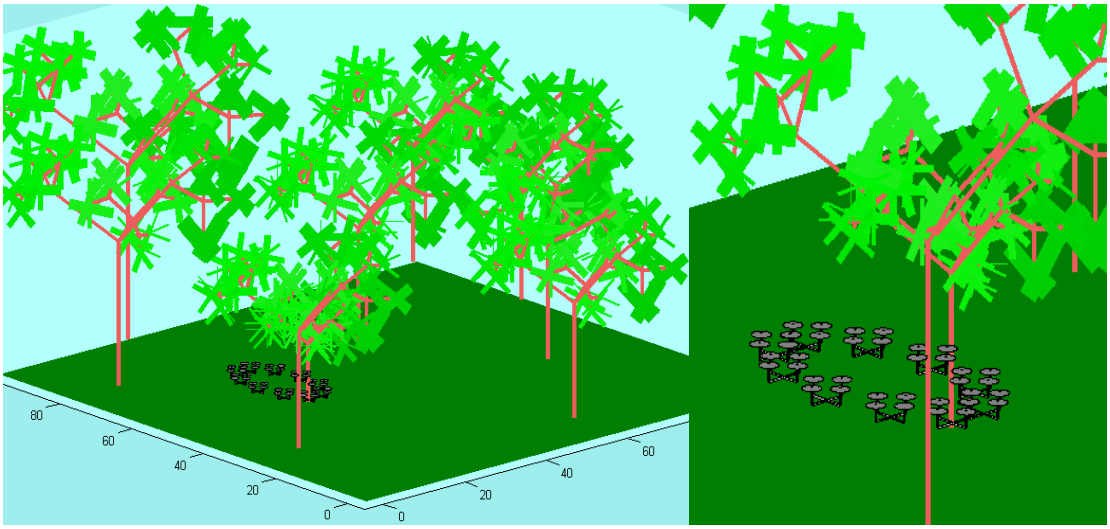


Fig. 2.17. Multi-agent quadrotor flight in formation across a forest.

Firstly, multi-agent formation flights require a fast and robust control system. This process allows the agents to maintain their positions whist transitioning between changing formation shapes. The control system is used to minimize the positional error for the quadrotors that are flying in formation. Most studies implement a double layer control system. The lower layer of the multi-agent control system holds the individual trajectory and mathematical model for each agent. On the other hand, the higher layer holds a controller that minimizes the position error of all the agents collectively. A robust control system is applied towards a group of homogeneous quadrotors in [91]. A two-level controller is applied. In this case, both the quadrotor and its formation control system are cascaded. They aim to minimize both the individual agent motion error as well as collective formation error. Research with a swarm of multi-agent quadrotors is performed by [92-93]. Their study aims to generate feasible flight paths for 20 quadrotors with extreme roll and pitch angles. The 20 quadrotors are further divided into smaller groups. The control system for each agent within a formation is

defined by their position error. The error function is a combination of an agent's local and global positional error. The local error defines the positional error of an agent within its group. The global error is defined by the agent's position within the entire group of 20 agents. Unlike spread missions, the minimization of each agent's positional error is the most important objective for any formation trajectory planner. It allows the agent to maintain their formation shape and avoid possible collisions.

There are two types of formation structures which are rigid and nonrigid shapes. In some studies, the agents are required to maintain a singular rigid structure across an entire trajectory. An example of a rigid formation structure is shown in Figure 2.18. In [94] a collective flight front must be maintained despite the introduction of communication noise. Consensus algorithm is used to define the network topology and implement cooperative control. The study shows that algorithm can keep a uniform front when maintaining noncomplex formation designs. It will require more time and constant communication to reach consensus with dynamic shapes. Missions such as payload lifting need rigid formation shapes. Here, the distances between each agent do not change with time. A system that allows agents to grasp and transport a payload is presented by [95]. Here, the transportation of four various shapes of wood planks is executed with quadrotors. The agents and its payload are mathematically modelled as a singular entity. This study applies a centralized control system to estimate the position, velocity and rotation of each quadrotor. It also uses a decentralized control system to predict the angular velocity of the agents. The combination of both centralized and decentralized control systems produces trajectories that are dynamically feasible. The estimated positional derivatives of each agent determine if the agent can maintain the rigid formation shape across its trajectory. It is highly important that the agents maintain their formation structure to successfully transport a payload.

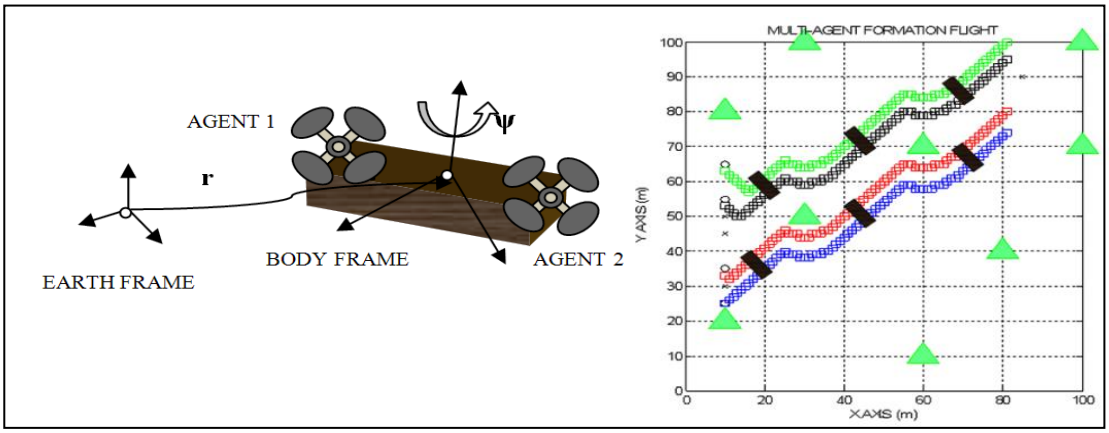


Fig. 2.18. Multi-agent quadrotor flight in formation across a forest with a payload.

Next, the design of adaptive formation structures allows the agents to change shapes when encountering narrow passages or obstacles. Tasks such as target tracking can be accomplished with nonrigid formation shapes. Nonrigid formation structures are more flexible and allow the agents to change shapes whilst flying. In [96] the formation topology for four quadrotors is defined through three different behaviours. The first

topology requires a constant network graph. The second topology is unconstrained and allows disconnection within the network. Lastly, the final topology allows shape changes whilst remaining connected. Whilst the study only performs two shape changes, it shows that a formation planner must be as flexible as possible. There are many formation shapes that are often used with quadrotors. Study [97] performs formation flying with three Qball-X4 quadrotors. Here, a variety of formation shapes such as line abreast, triangular as well as cross formation are tested. Results show that the quadrotors can achieve these shaped whilst satisfying Reynold's rules of flocking. Flocking behaviour is described as a swarm of agents that can maintain separation, alignment and cohesion whilst flying. The agents within a formation must avoid any possible collisions with neighbouring agents. All agents must also fly at the same velocity and direction of their fellow teammates. These rules can be used as guidelines for successful nonrigid formation planning.

Current works often test the ability of a multi-agent quadrotor system to adapt to different formation shapes across their planned trajectory. In this case, most objective functions minimize the number of shape changes and its complexity in terms of scale. The complexity of formation design is dependent on the difference between two consecutive formation shapes. The structure of the formation can either scale up or down in size. Research by [98] uses a team of three KMel K500 quadrotors to perform agile manoeuvring whilst maintaining their formation structure. The agents are set to transition from a line to a triangular formation shape and then returning to their initial shape once again. Their work highlights the importance of designing a planner that generates short and dynamically feasible trajectories that remain collision free whilst the agents transition between formation shapes. Analysis in [99] shows that the multi-agent quadrotors are capable of transitioning between many shapes. Here, the quadrotors progress from a tandem formation to an alongside, triangular and extended triangular structure. The study prioritizes objectives such as collision avoidance, good communication network, time optimality as well as minimal positional error. Due to this, the agents can maintain a less than 3m positional error despite strong winds. Both these studies show that there are many similarities between the spread and adaptive formation mission. Researchers still want paths that are short, collision free, dynamically feasible and energy efficient.

Another important factor that must be considered when designing paths for formation flights is the amount of time that is dedicated towards changing formation shapes. The rise time is defined by the time difference between two formation shapes. The initial time begins when the agents begin to change shape. The rise time ends when the agents successfully achieve the desired formation shape. Study [100] presents a trajectory planning and replanning algorithm for quadrotors. The authors highlight the importance of measuring the rise time of a trajectory. It is shown that if the rise time of a formation is small then the agents move too quickly and fail to track their desired trajectory accurately. The opposite is true for longer rise times where the flight time and fuel consumption are increased. Thus, it is important that the formation planner

strikes a balance when designing the formation trajectory and its control system. Research [101] presents a time-varying formation control system for five quadrotors that takes into consideration the different velocities of each agent. The quadrotor swarm can achieve the predefined time-varying formation structures through consensus-based formation control protocols. Here, the formation rise time can be estimated through the varying velocities of each agent. The five quadrotors can achieve the predefined time-varying formation within 200 seconds. This time-varying formation planner makes it possible for the agent to track the designed paths within a feasible rise time and minimal error. Similarly, study [102] aims to improve the formation rise time that is required by a group of multi-agent quadrotors. This is achieved by improving the convergence speed within a formation control system. In this case, the agents required less than 100 seconds to create a uniform front. This objective function measures the complexity of a formation trajectory. The longer the rise time, the more complex the formation shapes across the planned trajectory. Most researchers prefer trajectories that avoid all obstacles without the need for highly complex formation shapes.

These studies perform optimization of multiple objectives with an aggregated function. More flexibility and knowledge for the end user can be obtained when the focus is on the optimization of all objectives. Firstly, our study applies a fully decentralized control system. This control system runs in parallel through a multi-threaded processing unit. Thus, the multi-agent control system can run simultaneously. It minimizes the risks that come with a centralized system and reduces simulation time. Lastly, the equal optimization of all objectives is performed within this study through many-objectives optimization.

2.4 TRAJECTORY OPTIMIZATION FOR MULTIPLE AGENTS

In real-life scenarios, important decisions require the comparison of each choice's pros and cons. As humans, we analyse these options for their advantages and disadvantages. Then, we pick the option with minimal disadvantages. The minimization of negative criterions is the key to optimization. In certain cases, one may want a set of good options as opposed to a singular one. The ability to produce various options is advantageous because it gives the user additional flexibility. The end user can pick the best choice out of a few good options. This is where multi-objective optimization (MOO) performs best. As shown in Figure 2.19, there are various MOO algorithms that can be used for path planning. MOO algorithms can analyse high amounts of data in the form of many solutions. MOO evaluates the cost values of each solution at each generation. It then eliminates the weaker solutions through comparisons. As iterations progress, the number of optimal solutions increase through the weeding out of suboptimal solutions.

The most common variation of MOO is optimization through scalarization. Here, solutions are often ranked through a weighted sum equation. This equation is formed by merging multiple objective functions. The advantage of this method is its simplicity. The weighted sum equation is easily modifiable by the end user. The weight

values of each cost can be used to create bias within the ranking process. The negative aspect of this process is assigning these weight constants can be tough. Most users don't have preferences or a full understanding as to which objectives are to be given priority. These weight values can cause an unwanted bias within the search space. Thus, it leaves the end user with suboptimal solutions. The lexicographic method sorts the cost functions in order of importance. This process is known as the lexicographic order. Here, lower priority objectives are optimized if they do no negatively impact the higher priority objectives. The disadvantage here is similar to the weighted sum method. Prior knowledge of the importance of each cost function must be available. The first multi-objective genetic algorithm is the vector evaluated GA (VEGA). VEGA was implemented by Schaffer in 1984 [103]. In this case, optimization is achieved through the application of objective vectors. The entire search space is randomly divided into a few groups. The number of groups is the same as the number of objectives that are being optimized. The fitness of each solution subpopulation is optimized with differing objectives. Thus, VEGA produces the best solution for each objective since each solution group is optimized in one direction. VEGA fails to deliver when the user requires a solution that optimizes all objectives.

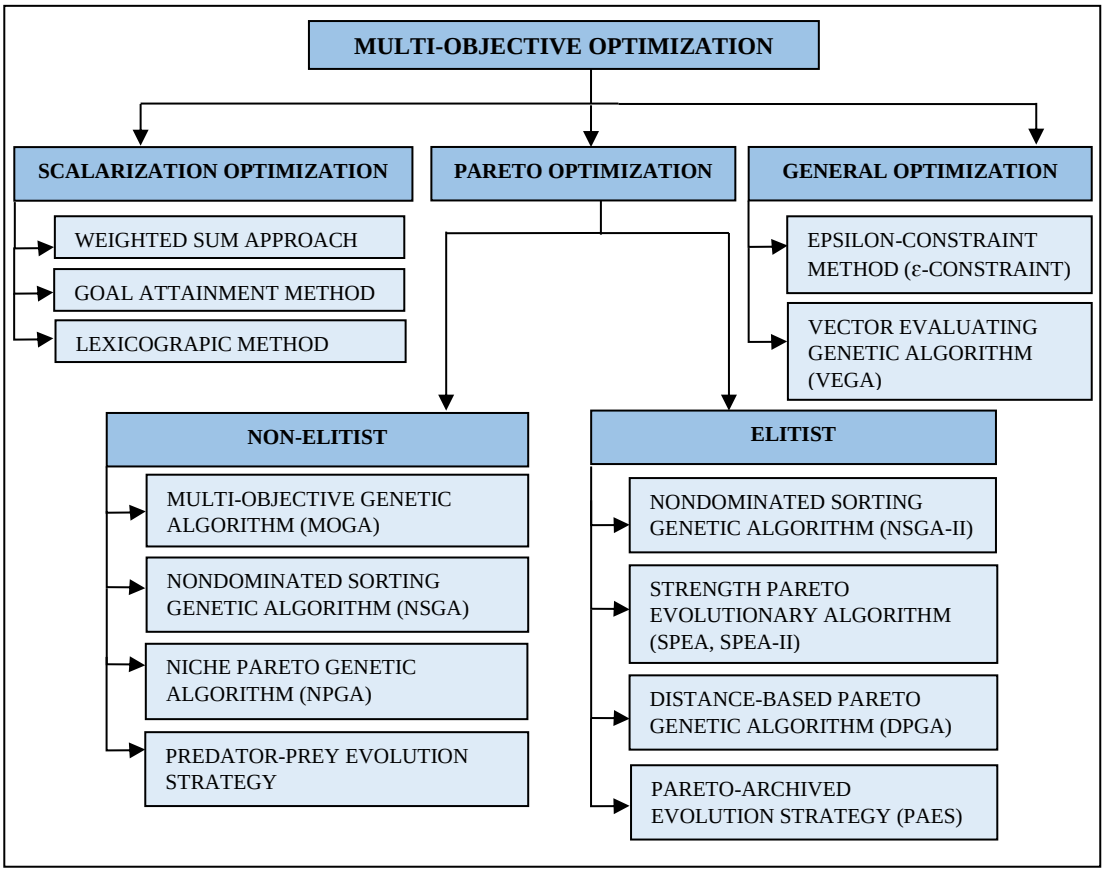


Fig. 2.19. Multi-objectives optimization algorithms

There are also optimization algorithms that apply Pareto dominance. Two terms that are often used within these algorithms are the Pareto Frontier and Pareto Optimal Solutions. Both terms are based on the following definitions:

- *Pareto optimal solution*. It is a solution where the values for all objectives are better than the solution that it dominates. These solutions are ranked as dominant/nondominated. A good collection of Pareto optimal solutions must be diverse and optimal.

- *Pareto frontier*. The curved Pareto front is formed by all the Pareto optimal solutions. These solutions form a graph that clearly defines a boundary between optimal and suboptimal solutions. The dimensionality of the Pareto front is dependent on the number of objective functions.

The approximation of the Pareto front can be highly beneficial to the end user. It allows the user to view solutions that are both diverse and optimal. The pros and cons of each option are also easily viewable. The multidimensional Pareto frontier is determined through the classifications of solutions. Here, solutions are ranked across the search space based on their Pareto optimality. These algorithms aim to maintain a balance between Pareto front convergence and solution diversity. Initially, Fleming and Fonseca implement multi-objective genetic algorithm (MOGA) in year 1993 [104]. The concept of fitness assignment through ranking introduced within their study. The rank of a solution is determined through the number of solutions that dominate it. Here, the minimal value of one is set for solutions that are never dominated. After dominance ranking, the level of diversity of a solution is tested through niching. The terms that are often used within the niching process are:

- *Niche radius*. Niche is used to define the spatial distribution of the solutions within the high dimensional space. The niche size defines the radius of similarity between solutions within objective space. The niche size dictates the probability of detecting a higher or lower number of optima.

- *Sharing function*. This function tests if a solution is within the niche radius of another. If the solution is further than the niche radius, it is given a value of zero. On the other hand, solutions that have many close-range neighbours have a higher sharing value.

- *Niche count*. The cumulative value of the sharing function for each solution is used to define the final niche count. The application of niche count defines if a solution's niche is crowded. It determines how many solutions are within a solution's niche radius.

- *Shared fitness*. The shared fitness of each solution is determined through the division of the raw fitness with the niche count. The shared fitness value ensures that solutions that are optimal but have many close-range neighbours within the

search are penalized. Thus, diverse solutions are maintained within the population until the next iteration.

These concepts are the building blocks for many GA based optimization programs such as Nondominated Sorting Genetic Algorithm (NSGA), Niche Pareto Genetic Algorithm (NPGA) and Strength Pareto Evolutionary Algorithm (SPEA). These algorithms differ in the way that the convergence and diversity mechanism are implemented. Srinivas and Deb [105] present a new fitness assignment scheme through NSGA. Here, nondominated solutions are ranked into different fronts across the search space. This algorithm evaluates the search space front by front as opposed. This has the benefit of avoiding early elimination of solutions which can occur when the entire population is evaluated as a whole. Each new front contains solutions that are less optimal than the prior front. The final fitness value of each solution is dependent on the front that it resides within. Solutions closest to the Pareto front have better fitness values. This procedure preserves solutions within each rank to create diversity. Next, solutions that are within the same front are tested for diversity. Similar to MOGA, a shared fitness value determines if a solution is maintained or discarded. Later, NPGA is proposed by Horn *et. al.* in 1994 as a non-elitist MOO [106]. This algorithm differs from the previous algorithm through its application of binary tournament selection method with raw fitness values. It reduces the processing time since it doesn't require the determination of shared fitness values.

Elitist algorithms such as NSGA-II and SPEA maintain a percentage of the best solution from the previous iterations for the selection process. In this case, the offspring are directly compared to their parents for survival of the fittest. NSGA-II also sorts the population into ranked fronts. It differs from the original NSGA because the total population is made up of both the parent and child population [107]. Despite its benefits, processing of the population is increased twofold with elitism. For diversity management, crowding distance is introduced. Here, the crowding tournament selection operator is applied for the creation of offspring. The crowding distance calculates the proximity of the solution's closest neighbours within the search space. The value of the crowding distance is obtained by drawing a virtual cuboid around the solution with its edges touching the nearest neighbours. The advantage of this algorithm is the removal of the niching variable which can be tough to determine.

SPEA was introduced in 1998 where an external population set is created out of the previously maintained elites. This external population is compared with the new generation elites [108]. This promotes the constant updating of the elite population as the generations go by. The algorithm differs from the others based on its strength fitness function and clustering method. The strength value is based on the number of solutions that a member of the elite population dominates. The cluster distance is determined through the Euclidean distance between all pairs of clusters. As SPEA progresses, clusters that are similar can be merged into one large cluster. Diversity is maintained by removing a representative solution from a cluster. The representative solution is

defined as the centroid of its cluster. As with the previous algorithm, SPEA eliminates the need for niche radius determination. The disadvantage of SPEA is the added complexity of managing both the external and current populations.

Each MOO differs from the other and can be applied towards various applications. The level of complexity and amount of processing time are key factors in deciding which multi-objective algorithm suits the optimization problem. Likewise, the Pareto frontier convergence and diversity are also important when choosing a suitable MOO algorithm.

2.4.1 MULTI-OBJECTIVES OPTIMIZATION

The MOO algorithms that are were discussed in the prior section has often been applied within studies that utilize quadrotor UAVs. This section analyses evolutionary algorithms that have been specifically used for path planning. Firstly, we look at the application of GA specifically towards trajectory generation. The process of path optimization requires a large population of trajectories. At each iteration, the algorithm performs a selection process that maintains optimal trajectories across future generations. In most cases, optimization is performed through an aggregated cost function. However, some studies have applied multi-objective optimization as well.

Genetic algorithm (GA) is used in [109] to obtain the best path for a quadrotor. Here, the cost function is a summation of node to node distance and a penalty value for close range obstacles. The application of GA without modifications required a large population of 500 members and 500 iterations to determine the fittest path. Thus, the basic algorithm would require a long run time as well as a large amount of data processing. Both GA and adaptive GA (AGA) are compared in [110]. This study generates paths for a quadrotor that is used for ground sensor detection. In this case, the adaptive crossover and mutation operators are dependent on the costs of each path. The authors show that these adaptive operators produce shorter trajectories in comparison to the basic GA. The disadvantage of this adaptive process comes in the form of added complexity. The user must be able to determine the operators accurately.

Work by [111] utilizes modified breeder genetic algorithm (BGA) with B-spline curves. This algorithm aims to generate paths across mountainous terrains. The authors consider different objectives with an aggregated weighted function. The number of costs applied is based on if the planner is operating online or offline. A balance between maintaining path diversity and optimality is done with an adaptive selection percentage. The feasibility of the generated paths cannot be guaranteed without the addition of the UAVs dynamics. Evolutionary algorithm (EA) for realistic scenarios is tested in [112] through the evaluation of 11 different objectives. This algorithm considers properties of real life UAVs, terrains, radars and missiles. In this case, different levels of priority are applied to the objective functions and constraints. As with the application of

weighted cost functions, the priority levels for each objective creates a bias within the population. This reduces chances of all costs being best optimized.

A comparison between the performance of parallel run particle swarm optimization (PSO) and GA for real-time path planning is researched in [113]. The application of parallel processing leads to a faster run time. Here, the algorithm performs migrations between subpopulations at every 10 generations. The authors conclude that GA outperforms PSO in terms of speedier convergence towards the first feasible trajectory. On the other hand, PSO produces more refined paths. NSGA-II and B-spline curves are applied within [114] towards offline path planning for multiple agents. Here, two planners for free flight and pre-specified flight points are designed. The multi-objective optimization algorithm is applied with dual conflicting objectives. The costs considered are the paths' length and height. The study shows that the extraction of global Pareto optimal solutions can be highly challenging even with a small number of objectives.

Building upon these prior studies, this thesis presents a modified version of genetic algorithm (GA). Here, GA is used with multi-agent RRT towards creating a large population of trajectories. In comparison to previously discussed works, this hybrid path planner is used in collaboration with a many-objective optimizer.

2.4.2 MANY-OBJECTIVES DOMINANCE AND DIVERSITY BALANCE

Real life flights require the consideration of many objectives simultaneously. Many studies choose to minimize the complexity of optimizing many objectives by prioritizing certain cost functions. In this case, it is important that all objectives are minimized or maintained without the extreme degradation of one cost over the other. In this thesis, many-objective optimization is applied to provide a diverse and optimal solution set for end users. The term many-objectives optimization is typically dedicated to the optimization of more than three objectives simultaneously that are often conflicting in nature. This algorithm is designed with an understanding that there is never just one optimal solution that is best in regard to all cost functions.

Many-objectives optimization is an expansion of the multi-objective optimization algorithms. They both vary in terms of additional mechanisms for balancing the Pareto front solutions diversity and convergence. The basic multi-objective algorithm is incapable of processing many objectives effectively. There are many challenges that can occur when the optimization of many objectives is treated as an extension of MOO. When implementing many-objectives, the main disadvantage is the loss of Pareto front convergence due to the large amount of nondominated solutions. This causes the algorithm to lose its ability to make adequate comparisons between the solutions during ranking. The second issue with optimizing many objectives is the number of solutions that are required to fully map the Pareto frontier. Higher number of objectives leads to a higher dimensionality Pareto front. Lastly, visualizing high

dimensional Pareto fronts can be highly challenging. This is especially true for frontiers that are higher than three dimensions. The Pareto front provides the end user with visual information regarding the trade-off values between each solution. Thus, providing the end user with equivalent data is important for the decision-making process.

Ideally, the population convergences towards the Pareto front with diverse solutions maintained. To achieve this, the basic MOO algorithm must be modified. Thus, the field of many-objectives optimization fulfils that need. Whilst being a new research area, a few studies have explored the potential application of many-objective optimization. There are researchers that have implemented many objectives within their optimization process. There are three methods that allow researcher to improve the selection pressure within their optimization algorithm. The options to choose from are as stated below,

1. Redefining the domination relation between solutions.
2. Modification of the diversity management system.
3. Dimensionality reduction.

There are many works that redefine the domination relation between solutions to increase selection pressure. The ranking is often improved with an aggregated weighted cost function [115]. This process merges the many objectives into a single function. As previously defined, the disadvantage of using weights is it creates bias during the optimizations process. Similarly, it also requires the end user to provide predetermined weight values. This can be challenging because the end user doesn't have prior knowledge of how much each cost function can be minimized. In many cases, its application comes at the cost of the deterioration of individual objectives. Another method for improving the ranking process is through the usage of fuzzy logic. Fuzzy-based dominance for many-objectives is undertaken by He *et. al* [116]. Progress in the fuzzification of Pareto optimality is shown in [117-118]. Here, the fuzzy set defines a broader spectrum of dominance between solutions. Typically, a solution is defined as dominant if it has better values for all objectives. In some studies, the sorting of solutions is performed by reducing the strictness of the ranking process. Here, solutions can be declared as dominant if most objectives are dominated. Selection pressure is also improved through alternative dominance criterions such as Pareto $\in$-dominance, α-dominance, *k*-dominance or preference weighting [119- 120]. Many of these studies improve on the ranking process by reducing the diversity of the solution set. This can be a disadvantage for end users that prefer a variety of solutions.

Some researchers modify the typical diversity management design to increase the selection pressure during the ranking process. Popular diversity operators are such as niching [121], crowding distance [122], clustering [123] or *k*-th nearest neighbour [124]. Adra and Fleming [125] apply a diversity management operator (DMO) as well as an adaptive mutation operator. They reduce the impact of both the operators with dynamic activation and deactivation of the diversity mechanism. This process can be

complex because the system must be capable of accurately determining when the diversity mechanism needs to be shut off. Recently, shift-based density estimation (SDE) strategy [126] is employed where the shifting of solutions based on its convergence value. In a pursuit to redefine both the dominance relation and diversity maintenance in many-objectives optimizations, [127-128], Zou *et. al.*[129] demonstrate the application of L-optimality towards a new definition of the fitness function that applies the principle of the minimal free energy in thermodynamics. Whilst the modification proves promising, the assessment of the algorithm is limited to less than ten objectives due to the computing power necessary for the resulting hypervolume. Similarly, [130] applied a grid based criterions [131] [132] to efficiently create a difference between each solution thus defining grid ranking and grid crowding distance for qualitative comparisons. The issue with redefining or minimizing the effects of the diversity mechanisms is that the Pareto frontier is often partially mapped. Thus, the end user is left with a set of representative solutions instead of a well spread Pareto optimal solution set.

This thesis applies Many-Objectives Optimization for the optimization of multi-agents and its objectives. The three challenges of applying many objectives are dealt within this research. This study aims to strike a balance between diverse and optimal solutions through dimensionality reduction. Here, the end user is supplied with high resolution visual imagery as well as organized data. The additional knowledge will assist the end user in making a final choice.

2.4.3 MANY-OBJECTIVES DIMENSIONALITY REDUCTION

Many studies have chosen to perform dimensionality reduction or sorting of the many objective functions. This is performed through objective clustering, preference-inspired approaches and the application of corner sort. Corner sort is implemented by [133] and [134] to minimize the complexity of many objectives optimization. Minimal objective comparisons are necessary to determine dominance when only the corner solutions of the Pareto front are compared. The disadvantage of this process is it requires the accurate identification of corner solutions. This can be challenging with high dimensional Pareto fronts that have a large number of objectives.

The next option for implementing dimensionality reduction is through the creation of objective subsets. Preference-inspired Co-EA (PICEA) is used in [135] to optimize preference based objective subsets. Here, the fitness of each solution is determined through the evaluation of both the full set and current objective subset. The combination of these sets promotes convergence towards an optimal final population. The usage of objective subsets is presented in [136]. These subsets contain random combinations of the full objective set. These subsets are applied in rotation across a constant number of generations. Both these papers aim to increase the selection pressure through merging of both local and global dominance. The disadvantage of randomly created subsets is it can lead to less efficient use of processing time. This is because the

comparison of nonconflicting objectives can be redundant when determining dominant solutions.

Another popular option for increasing selection pressure is through objective reduction. Objectives that are not crucial towards the mapping of the Pareto front are deemed unnecessary. Objective clustering is applied in [137]. The deletion of redundant objectives is performed within clusters that hold nonconflicting objectives. Study [138] identifies redundant objectives through eigenvalue and correlation matrix analysis. Objectives that are found to be nonconflicting are removed from the full objective set. Both studies are performed under the assumption that the determination of redundant objectives is highly accurate. Unfortunately, this leaves no room for error in cases where a nonredundant objective is eliminated.

In this research, we apply dimensionality reduction to increase selection pressure without the absolute removal of any objectives. Here, the objective subsets are not created randomly. It is performed by evaluating the level of conflict between objective pairs within each subset. This process minimizes the chances of full elimination of an objective function and leaves room for possible error.

2.5 SUMMARY

This chapter presents studies, researches and inventions that have inspired the design of the multi-agent quadrotor's optimal path planning algorithm. Firstly, the development of the quadrotor is discussed. It shows the various sizes and types that are available within the current market. It also defines the large number of missions and applications that use the quadrotors. The missions have different objectives and test environments. All of this information shows that the quadrotor can be extremely robust and flexible. It is the ideal vehicle for a multi-agent UAV system.

Next, the two main applications that require a multi-agent UAV system are defined. The first is the spatially spread flight scenario where the quadrotors are required to fly independently within a team. They must collaborate to successfully complete missions such as reconnaissance or search and rescue. The second application that is presented is multi-agent formation flight with quadrotors. In this case, the trajectories of each agent are highly dependent on each other. Both applications utilize similar path planning algorithms to generate feasible paths. Here, sampling-based path planning algorithms such as Rapidly Exploring Randomized Trees are defined in detail. The last section presents the various multi-objective and many-objective optimization algorithms. Focus is placed upon both Genetic Algorithm and Dimensionality Reduction Many Objective Optimization.

The literature review that is presented will be used as an inspiration for the multi-agent quadrotor optimal trajectory planner within this study. The next chapter presents the Multi-Agent Rapidly Exploring Randomized Forest (MA-RRF) and GA

hybridized path planners. Firstly, the three test environments are presented. The various challenges that exist in each test space are discussed. The MA-RRF algorithm is then used to map the test environments with speed. Finally, initial feasible paths for GA of the four quadrotors are shown. These paths will form the initial population for the optimization process. Lastly, the mathematical model and the control system for all agents are described in detail.

CHAPTER 3: MULTI-AGENT QUADROTOR TRAJECTORY PLANNING

A path planning algorithm typically performs two things which are mapping the free space within an environment and path extraction. Three test environments that pose different challenges are presented in this chapter. Each test space will test the ability of the trajectory planning algorithm to design paths across a variety of environments.

This study aims to fill the gap of planning paths across long range environments with minimal moving obstacles. All of these environments are randomly generated and different for each experiment. They are designed to imitate the major structures within their real-life versions. These environments only simulate static obstacles. Despite the exclusion of dynamic obstacles, this algorithm is applicable in disaster zones that contain many static obstacles. One example is war zones with abandoned buildings or toxic chemicals. Another example is the to perform surveillance above rainforests at high altitudes above the trees. These environments may contain some dynamic obstacles that can be avoided with the usage of an ultrasonic sensor. The agents can perform a swift obstacle avoidance turns and return to their planned path.

Firstly, a high-rise cityscape that possesses maze like narrow passages is shown. Next, a highly cluttered indoor environment is developed. This space simulates real life rooms with windows and doors. Each room contains hardware and furniture at variant heights. The clutter in each room is modelled through randomly placed cubes. Lastly, a mountainous terrain that has sudden and gradual terrain height changes is displayed. This space will test the capabilities of the path planner in terms of obstacle avoidance.

This chapter also presents two path planners that will collectively generate nodes for the multi-agent trajectories. Both path planners are designed to plan hybridized paths. Initially, the path nodes for each quadrotor are generated through the Multi-Agent Rapidly Exploring Randomized Forest (MA-RRF) planner. Here, both free space mapping and path extraction are performed. Each MA-RRF path hybridizes branches from different trees and forest links. Thus, a path is a hybrid of many tree branches. Later, the suboptimal trajectories that were designed by MA-RRF will form the initial path population for the many-objectives optimization algorithm. New paths for the next generations are constructed through the crossover and mutation process within Genetic Algorithm (GA). These paths will be hybridizing different sections of the initial MA-RRF paths in order to create new path populations. This means a path by GA is a mesh of the parent paths by MA-RRF.

The final step for successful path planning is the path repair process. This is

necessary because the GA operators often create path subsections that collide with obstacles. In this study, the MA-RRF algorithm is reapplied within GA to repair the new generation paths. The planner quickly maps the obstacle and generates a feasible route between the collision points. These new paths will undergo multiple post processing steps in order to maintain the level of diversity within the population. The child paths that survive the filtration process will be combined with their parents. These paths will form the current generation of multi-agent quadrotor paths.

Lastly, this chapter defines the mathematical model and control system of a quadrotor UAV. These two subsections will form the closed-loop control system for the agents. The previously designed trajectories will be used as the input for the quadrotor UAV control system. The information that is generated by the control system will be used to predict the values of the many objectives that are presented within Chapter 4 of this thesis. These cost estimations will then be used in Chapter 5 to slowly filter out suboptimal or non-diverse trajectories.

3.1. THREE-DIMENSIONAL ENVIRONMENT FREE SPACE MAPPING

In this study, two methods of free space mapping are utilized. Both a sampling-based planner and grid blocks are used to map the simulated environments. In this study, only the mountainous terrain is defined through grid blocks. This is because the small and gradual changes within the environment can be computationally exhaustive to accurately define. All of the other test environments do not require grid free space mapping during the path planning process. A quick sampling-based planner is used to map and spread nodes across the test spaces. In this case, the obstacles and free space do not need to be well defined prior to the sampling process. The parameters for both the free space mapping processes are shown in Table 3.1. The obstacles within all test spaces have buffer regions that are placed around them. These boundaries will reduce the possibilities of obstacle collisions.

3.1.1. HIGH RISE CITYSCAPE ENVIRONMENT

Cityscapes around the world are urban spaces that are made up of many buildings. Drones are commonly used for surveillance or filming important social events. They are typically used by photographers and journalist to document newsworthy events. This creates a large pool of real-time data that can be instantly shared between the local people. The UAVs can be applied for crowd and human traffic management during natural disasters. In recent times, multi-agent quadrotors have also been used for payload delivery. Companies such as Amazon have attempted to perform air delivery service for their customers. Many entrepreneurs have opened small businesses within the city to manufacture and test quadrotors. These drone hardware companies will continue to create

TABLE 3.1. SIMULATION ENVIRONMENT PARAMETERS

Environment	Description	Value
Cityscape	Test Space Size	230 x 230 x 200 m^3
	Grid Size	15 x 15 x 15 m^3
	Number of Skyscrapers	18
	MA-RRF Buffer Region	5 *m*
	Safety Zone Boundary	6 *m*
	Obstacle Sampling Distance	5 *m*
Highly Cluttered Indoor	Test Space Size	120 x 120 x 120 m^3
	Grid Size	15 x 15 x 15 m^3
	Number of Rooms	5
	Number of Random Sized Cubes/Room	6
	Number of Windows	5
	Number of Doors	5
	MA-RRF Buffer Region	1.5 *m*
	Safety Zone Boundary	6 *m*
	Obstacle Sampling Distance	2.5 *m*
Mountainous Terrain	Test Space Size	65 x 65 x 70 m^3
	Grid Size	15 x 15 x 15 m^3
	Size of Peak Obstacles	8 x 8 x corner peak height m^3
	MA-RRF Buffer Region	1.5 *m*
	Safety Zone Boundary	6 *m*
	Obstacle Sampling Distance	4 *m*

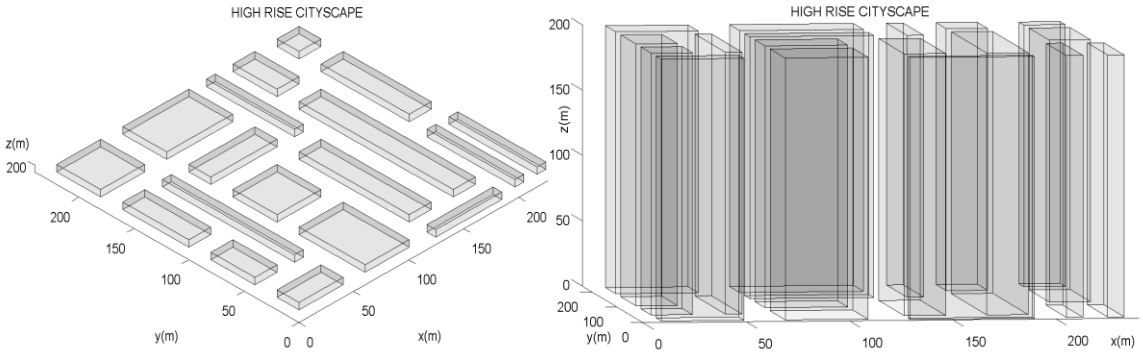


Fig. 3.1. Top and side perspective of a cityscape environment.

new jobs within their cities. Thus, quadrotors are no longer an unfamiliar sight for city folk. Many cities have introduced their own drone-related regulations in order to protect the privacy of their citizens. The quadrotors can continue to operate within a city as long as the users comply with these rules. Both leaders and regular citizens are often in conversation about the impact of deploying drones across a city. The popularity of the quadrotor within cities shows that a path planner must be capable of designing trajectories across high rise cityscapes.

Most cities around the world have a few common characteristics. It is important to define the structures that collectively create a cityscape. The most visible section of any

urban space is its buildings. As previously defined, many people live or travel daily into the city for job opportunities. Similarly, youth and children often come to the city to study. Families come together to enjoy the many restaurants or any forms of entertainment. Thus, the buildings within a city are made up of large corporations, small sized offices, shops, hospitals and schools. These structures can be built with different shapes, sizes and height. Still, most buildings are square and simple in design. Next, there are all types of clutter within an urban space. There are street lamps, plants and vehicles that are placed on the ground. Buildings regularly have billboards, lights or a balcony attached to them as well. Another defining characteristic of a developed city is its roads. Urban spaces are often filled with large crowds and heavy road traffic. The streets are travelled by people on foot or different modes of transportation. The roads within cities are unlike those in underdeveloped areas. City roads are narrow and have many sudden turns. It moves across tight spaces between buildings. All of these characteristics show that there is a lot of activity that occurs at the ground level of a city. It is important that the multi-agent quadrotors fly above crowds and around buildings.

This study recreates this test space with a simulated three-dimensional cityscape environment as shown in Figure 3.1. The clutter that is typically placed within an urban environment isn't included within this simulation. The purpose of the cityscape environment is to test the path planner's ability to plan paths within long range narrow passages. The planner aims to create well minimized trajectories where sharp cornered turns exist between city buildings. Large rectangular cubes form the buildings within the simulated high-rise cityscape test space. These cubes are spaced closely to one another across the environment. This environment is not made of buildings with various heights. Based on prior experimentation, the path planner will seek nodes that exist above the buildings. This is because the spaces above buildings are less constrained than the narrow passages between them. Thus, this simplifies the challenges of navigating across real life high rise cityscape environments. Here, all the buildings are maintained at a similar height in order to force the path planner to fly between buildings as opposed to just above buildings. This creates an extremely constrained environment that challenges the trajectory planner in a way that the other environments do not. The agents must fly at higher heights and be capable of avoiding the corners of each building. The agents must also be capable of undertaking extreme manoeuvres as they fly from one road to another.

Next, mapping the free space within the cityscape is paramount to creating paths that are collision free and diverse. The boundaries for this environment are the most simplistic out of all the simulated environments. This is due to the unvarying heights and width of the buildings. It can be tough to determine the size of the boundaries around each building. If the boundary is placed at a large distance from the building, it promotes safer turns for the quadrotors. The downside to this is larger boundaries can cause the narrow passages to become too tight for many agents to travel at once. It also reduces the

number of diverse paths because it limits the spread of the sample points. On the other hand, boundaries that are placed too close to their building can be dangerous for the multi-agents. This increases the chances of possible collisions especially at sharp corners. Thus, the size of the boundary should strike a balance between the two extremes. After the free space mapping is complete, the sampling of the obstacles is performed. The sample points aid in defining the formation shapes across each trajectory. Three-dimensional high-resolution sampling is not necessary for multi-agent formation flights. As shown in Figure 3.2, obstacle sampling is only performed across the roof of each building. Here, the distance between the sample points and path nodes can provide sufficient information for the formation flight planner.

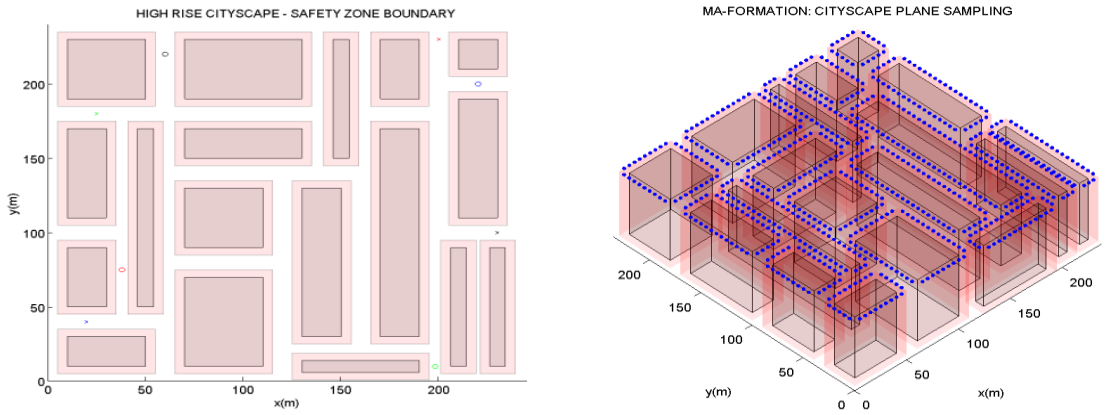


Fig. 3.2. Cityscape environment's safety boundaries and boundary plane sampling.

3.1.2. HIGHLY CLUTTERED INDOOR ENVIRONMENT

The second test space as shown in Figure 3.3 is a highly cluttered indoor environment. This simulated environment mimics spaces such as domestic residences or employment venues. This environment is extremely different from the other two test spaces because it is placed within the interior of a building. Firstly, the largest consumer market is made up of hobbyists. One variety of a quadrotor that is used indoor is the Parrot AR. Drone that is controlled by Apple products. The drone comes equipped with a high-resolution camera that allows for real-time recording and sharing. Family parties and ceremonies can be instantly recorded. It connects to an indoor wireless connection and uploads flight information online. This process creates a community of users that share a common hobby. The second group of people are business owners that are manufacturing their own quadrotors. Initial testing of the quadrotor is often performed indoors. Lastly, multi-agent quadrotors are often flown by researchers within their laboratories. These experiments are safely conducted with nets enclosing the agents inside it. Thus, there are a large number of users that fly quadrotors within indoor spaces. This shows that a path planner must consider the challenges that occur within the interior environments.

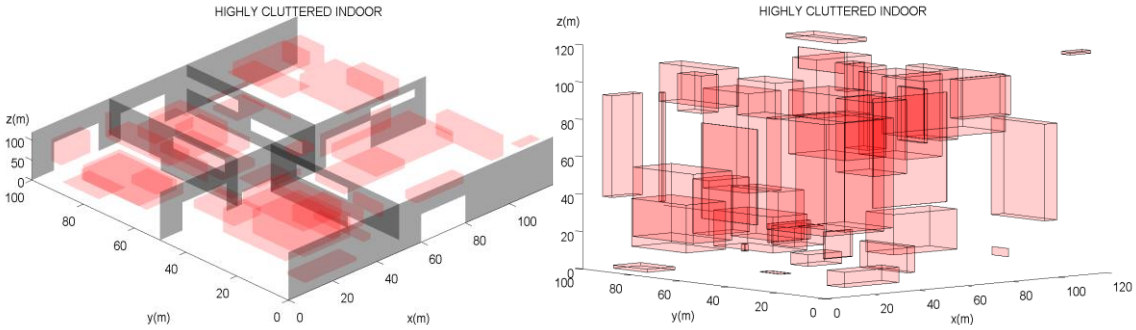


Fig. 3.3. Top and side perspective of a highly cluttered indoor environment.

Similar to cities, indoor environments have a few common characteristics. Identifying these characteristics is important for designing a realistic indoor test space. There are different obstacles within this space. Interior spaces are made up of walls, windows, doors, furniture and electronics. Most rooms have a large amount of clutter that are scattered at various heights. The clutter is a representation of the typical ground furniture, high level shelves, hanging light fixtures, gadgets and electronic equipment. It also includes other decor objects. All of these objects are constrained within extremely small spaces that are used by humans daily. Thus, the trajectory planner must be capable of obstacle avoidance. This is achieved by planning paths that move above and below all clutter. It forces the agents to perform aggressive manoeuvres around its obstacles. Another challenge that this environment poses is its constricted entry points. These entry points are either doors or windows that allow the quadrotors to transition from one area to another. It is highly challenging to detect entry points across large walls. If the planner fails to accurately detect these doors and windows, the rooms in between them will be neglected in the path planning and optimization process.

This research recreates this test space with a simulated three-dimensional highly cluttered indoor environment as shown in Figure 3.3. The walls are pieced together with gaps that represent the doors and windows. Each window has a different size. Many homes have pets that live with their owners indoors. Here, entry ways for pets are also included. The clutter within rooms is simulated through various sized cubes. These cubes are randomly generated across each room. The end user has the option of introducing more clutter in order to test the capabilities of the path planner. Next, the obstacles within the simulated indoor environment have boundaries placed around them. Table 3.1 shows that the boundaries are placed much closer to the obstacles in comparison to the cityscape environment. This is because the entry points for each room are already small in size. Boundaries that stick out too much can close up the entry point. It also stops the agents from transitioning between rooms. Thus, a balance must be struck between safety and space exploration.

Lastly, the obstacles are sampled to provide markers for the formation planner. As shown in Figure 3.4, the sampling process is more complex than the cityscape environment. This test space will require varied height sampling to properly define the formation shapes across a trajectory. It also requires a higher sampling rate. Sampling is performed on the top portion of each wall and clutter block. The sampled planes also define the height of the obstacles. They are only a threat to the flight trajectory node if the waypoint is within the lower and upper height limit of the obstacle blocks. The free space contours around these obstacles can be extracted by determining if the path waypoint is above or below any obstacle.

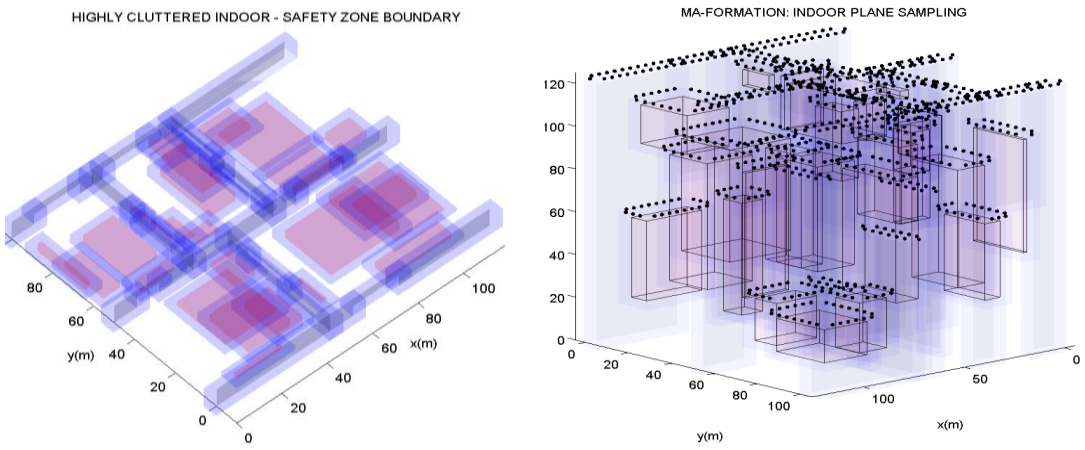


Fig. 3.4. Highly cluttered indoor environment's safety boundaries and plane sampling.

3.1.3. MOUNTAINOUS TERRAIN

The last terrain that is generated by fractals is the mountainous terrain as shown in Figure 3.5. Quadrotors are often used across mountainous terrain for a variety of reasons. As with the prior test environments, many wildlife photographers and journalists use the quadrotor to capture valuable data. National Geographic photographers have used quadrotors for many years. These aerial vehicles are flown across Borneo rainforests, Serengeti ecosystem and volcanoes. In some cases, the quadrotor is used to capture imagery of the wildlife within the mountainous terrain. Next, the vehicle is also used for search and rescue. There have been emergency situations that have occurred within forestations such as fires or landslides. The quadrotor is capable of providing real time imagery to the rescue team. Lastly, the multirotor system is also deployed across mountains to evaluate environmental issues such as deforestation or destruction of animal habitats. It allows the law makers to create rules that minimize extreme climate changes and labour abuses. All of these applications prove that it would be advantageous to have a path planning system that is capable of navigating across mountainous terrains.

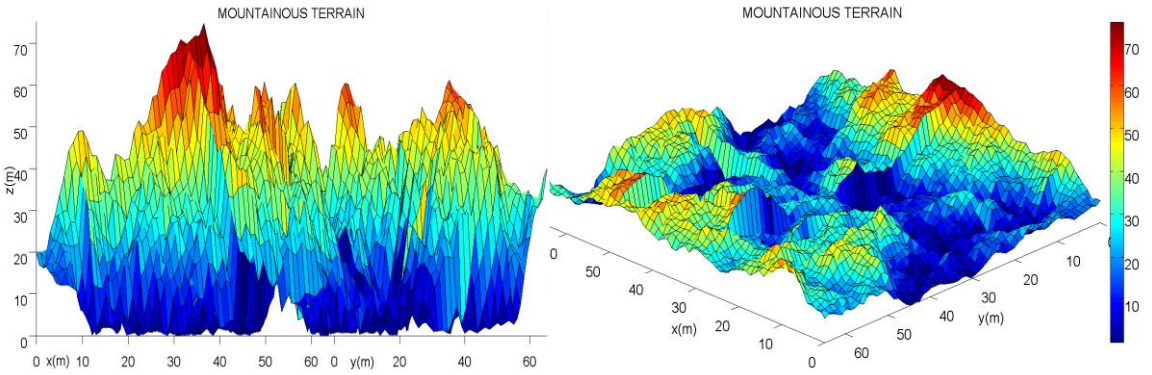


Fig. 3.5. Top and side perspective of a mountainous terrain.

There are many key characteristics that define a mountainous terrain. The simulated test space is designed based on these characteristics. It is an environment that is filled with trees of different heights. The trees can grow tall and create a dense forest. There are smaller sized plants such as ferns and flowering shrubs. It also has rivers and pathways that progress across the terrain. With most mountains, the changes in height occur gradually from the ground level. Similarly, in this study, the terrain isn't filled with many sudden high peaks. Here, there aren't any extreme height differences between each peak. Figure 3.5 shows gradual height differences across the environment. Focus is placed on generating trajectories between the peaks instead. This is because there aren't many researchers that have flown the quadrotor at extremely high altitudes. Quadrotor UAVs do not have adequate fuel to make trips that are long range and require a high vertical climb. This may change in the future when batteries have a longer discharge rate.

The free space mapping process of the mountainous terrain is more complex than the prior two test spaces. Here, high resolution free space mapping is necessary for efficient obstacle avoidance and formation planning. The obstacles within this terrain are mapped with high accuracy. The mountainous terrain is unlike the cityscape and indoor environment where boundary planes are easily placed around each obstacle. The gradual height differences across the mountainous terrain are difficult to capture if there isn't any separation between them. Due to this, the path planner can't design path that allow the agents to fly close to the peaks. This reduces the chances of finding the shortest path. Firstly, the peaks within the terrain are divided into equal sized obstacles. This process creates a large number of smaller sized obstacles within the terrain. The planner must strike a balance between accurate terrain free space mapping and processing time. The height of each obstacle is defined based on the mountain peaks that are within it. Next, safety boundaries are places across all obstacles. Lastly, sample points are spread across the top plane of the safety boundaries as shown in Figure 3.6. These sample points will assist the formation planner in extracting the free space around the path nodes.

This environment requires the multi-agent quadrotors to fly collectively whilst avoiding peaks. The mountainous terrain's exploration is limited to 90% of the highest peak. This is to avoid flight over all peaks and force the algorithm to plot paths between peaks. The quadrotors are required to fly through sudden height changes in terrain in the form of peaks and lows. It is important that the agents successfully avoid local minima and reach their destination. The multi-agent trajectories within this final environment are simplistic in nature. There aren't many sharp corners or narrow passages. There is a lot freer space than the prior two test spaces. The challenging aspect of this terrain is the high resolution free space mapping that is required. Thus, the path planner is required to generate a diverse set of paths that do not cause any possible collisions with the mountain peaks.

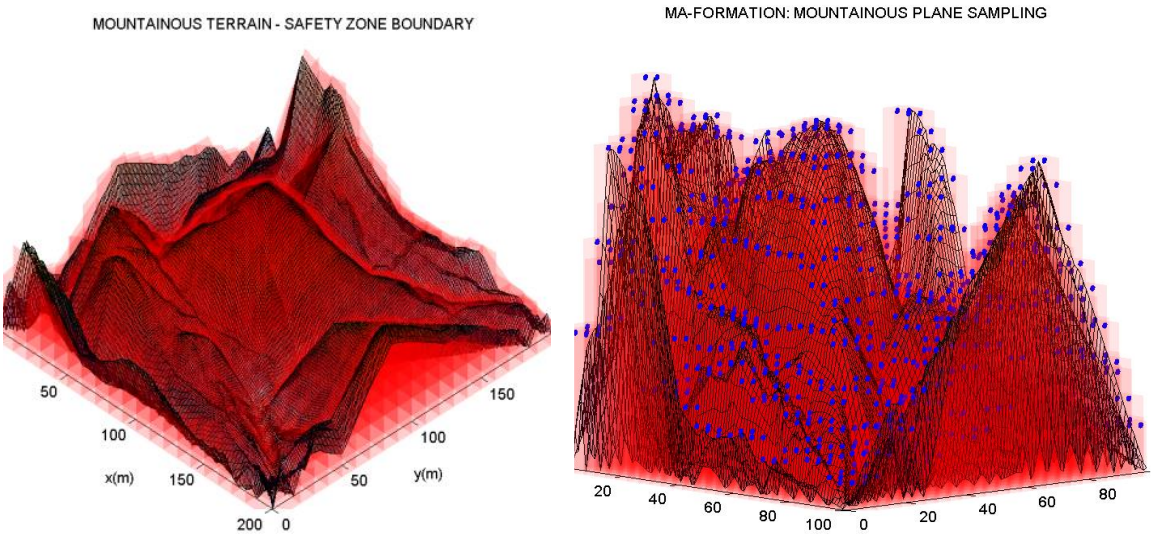


Fig. 3.6. Mountainous terrain's safety boundaries and plane sampling.

3.2 MULTI-AGENT RAPIDLY EXPLORING RANDOMISED FOREST

Numerous studies apply sampling based algorithms such as rapidly exploring random trees (RRT) because it is able to promote complete coverage with speed. In this research, we apply modified versions of RRT path planner, GA and many-objectives optimization. The optimization process requires a large population of trajectories at every generation. Here, both MA-RRF and GA are used to supply the many-objectives optimization algorithm with a diverse set of multi-agent trajectories. Each subsection has been modified to suit and benefit from a multi-agent system. As previously defined, this combination produces an algorithm that creates a final population of diverse and well minimized trajectories for all quadrotors.

Firstly, initial path waypoint generation is performed through a hybridized version of the RRT algorithm. RRT is used because GA requires a larger amount of time to fully

explore an environment. RRT is capable of free space mapping and extracting paths at a much faster rate. On the other hand, GA is capable of merging a few good paths to create a larger collection of good paths. The combination of both algorithms reduces free space mapping time and maintains good paths across generations. The MA-RRF planner generates the initial population of paths for the two-stage path planning algorithm. The second stage applies a modified GA to produce a diverse population of trajectories across all iterations. Here, GA hybridizes different subsections of the initial MA-RRF paths to create new generation paths. MA-RRF is also reapplied for the path repair of child trajectories after the crossover and mutation process within GA.

This version of the basic RRT planner aims to generate a large collection of paths for each agent simultaneously. With a multi-agent system, multiple trees can be generated to further speed up the exploration process. Thus, the MA-RRF trajectory planner is designed to fully harness the advantages of having a multi-agent system. It is also able to function within all types of environments. The MA-RRF algorithm is tested across three environments that have a variety of challenges.

3.2.1 RAPIDLY EXPLORING RANDOMISED FOREST

The application of RRT is based on its ability for quick exploration of large unexplored areas of space despite high amounts of obstacles. This lends to the fast extraction of feasible paths within high-dimensional free space for vehicles with high degrees of freedom such as the quadrotor. Most studies that apply the RRT planner typically use a single tree [31].

The standard RRT algorithm is defined below. The algorithm begins by defining the root of its tree at the initial position of the UAV, $x_{init} \in X_{free}$. Next, a sample point, x_{rand} is placed randomly across the test space, X_{free}. This sample point is connected to the root of the tree if there is no obstacle between them. An input, u is applied to connect and minimize the distance between the two nodes. This connection creates the first tree branch, x_{new}. As the tree grows, the random sample points are then connected to the nearest collision free branch, x_{near} on the tree, T. The two nodes that create each tree branch are stored within the database as a parent and child node. The input, u is also stored within the database as well. The end point of the branch is defined as the child node. This relationship simplifies the extraction process of the final path. The free space mapping process continues until a termination point is met. The termination point is often defined by the tree's closeness to the goal node. Thus, the RRT algorithm is stopped when a tree branch is within close range of the UAV's destination. Lastly, a feasible path is obtained by extracting the parent nodes of all connecting branches in reverse until reaching the initial node.

There are a few variations of the standard RRT planner. The most popular is the single-tree search. This program can be improved by introducing a cost function that creates bias towards the goal node. Thus, the sample nodes connect to tree branches that are near in distance and also close to the goal node. The next option is a balanced bidirectional search. Here, two trees are utilized in an attempt to reduce run time. One is rooted at the UAVs start node whereas the other is placed at the goal node. The algorithm constantly attempts to connect these two trees. After a connection is made, the path can be extracted from the branches of both trees.

STANDARD ALGORITHM Rapidly-Exploring Random Trees

Input: Initial state x_{init}

Output: RRT graph T with K number of vertices

GENERATE_RRT ($x_{init}, K, \Delta t$)

1: $T.init\ (x_{init})$;

2: for $k = 1$ to K do

3: $x_{rand} \leftarrow$ RANDOM_STATE ( ) ;

4: $x_{near} \leftarrow$ NEAREST_NEIGHBOR (x_{rand}, T)

5: $u \leftarrow$ SELECT_INPUT (x_{rand}, x_{near})

6: $x_{new} \leftarrow$ NEW_STATE ($x_{near}, u, \Delta t$)

7: $\mathrm{T}.add\_vertex\ (x_{new})$

8: $\mathrm{T}.add\_edge\ (x_{near}, x_{new}, u)$

9: Return T

Lastly, more than two trees can be generated with a multi-tree RRT system. There is an added complexity with multi-tree systems. The additional trees can be introduced at the beginning or in between the simulation process. The designer can choose to create more trees in locations that are tough to reach. Expanding the standard RRT into multiple individual RRT trees still causes an increase in processing time. The algorithm must continue to expand these trees whilst searching for possible connections between them. Thus, these connections must be made strategically in order to minimize complexity. Similarly, each tree connector must be stored properly within the database. The path extraction process will be highly challenging since the path subsections are made up of many different trees. In order to improve the basic algorithm, MA-RRF is presented as an alternative. This study implements a multi-tree program that generates a large collection of diverse paths.

3.2.2 MA-RRF PATH PLANNING

This section presents a sampling based planner for solving multi-agent path planning problems in high-dimensional configuration spaces. Here, MA-RRF is applied in an attempt to generate a large collection of feasible paths for each agent simultaneously. This multi-tree system tries to maintain a balance between space exploration and processing speed. There aren't many works that have implemented a multi-tree sampling based planner for multi-agent UAV systems. This study aims to improve on the basic multi-tree RRT free space mapping algorithm by strategically placing links across the individual agent's trees. These tree-to-tree links are used to merge the tress into a full forest, MA-RRF. In this case, not all collision free links are explored. The algorithm attempts to create forest links that are diverse in location. There also shouldn't be too many or too little links between different agent's trees. This increases the possibility of obtaining a diverse collection of paths.

The second challenge of coordinating a multi-tree system is creating a storage system that allows the end user to comprehend the progression of each forest branch. This storage system must also minimize the time required to extract the final paths for the multi-agent system. In this research, a mutual database is constructed where data from the free space mapping and multi-agent paths are stored. This shared database is used by all agents. Thus, it allows the agents to have access to a shared database that is constantly updated by their neighbouring agents. Many variables are systematically stored within the database. These variables include the relationship between each parent and child branch of each tree. It also contains the relationship of the open tree branches that create the forest links. Due to this, duplicate branches and unnecessary links are easily avoidable. The nodes for each path can be located across many trees. This shared database will allow the algorithm to recursively extract each path subsection. This increases efficiency and reduces the complexity of running a multi-agent quadrotor path planner.

The full Multi-Agent Rapidly Exploring Random Forest is shown in Algorithm I-IV. This research tests the algorithm with four agents, N_A that are spread across each test space. The building of the MA-RRF begins by defining the initial location of each quadrotor. As previously defined, this work presents two multi-agent applications which is MA-Spread and MA-Formation. The first application requires a collection of paths for multiple agents that are flying independently. Thus, the MA-RRF algorithm is applied as defined in Algorithm I. On the other hand, the second application only requires a collection of reference paths. This is similar to path planning for one agent. The end user has the option of using the multi-tree system for one agent. Virtual agents can be placed around the test spaces to define the root of each tree. In this study, the database of paths that have been generated for the MA-Spread scenario is reapplied for the MA-Formation application. Here, the paths that were generated for the first agent are used as the reference path for the formation flight.

This work assumes the the environment is a known, randomly simulated, test environment with static obstacles. It also assumes that the end user requires multi-agent paths for long range flights. The chosen paths will be broadcasted to the agents after the optimization process. The initial free space mapping that is performed across each test space is applied as the input for the free space mapping algorithm. Here, the safety regions around each obstacles is defined as a no fly zone. The MA-RRF planner allows the end user to define the level of diversity that is required for the initial population of the GA. This is set through the similarity threshold, $S_{threshold}$. Next, each tree is rooted at the start, $\rho_i(t_0)$ and the goal configurations, $\rho_i(t_f)$ of each agent. These roots are used to expand the individual trees for the first iteration.

ALGORITHM I MA-RRF Path Planning Algorithm

Input: Number of agents, $i = \{1, \ldots\ldots, N_A\}$

Updated multi-agent obstacles database, ξ_{obs}

Number of iterations/sample points, t_f

Agent's current state , $\rho_i(t_0) = \{x(t_0) \quad \dot{x}(t_0) \quad \ddot{x}(t_0)\}$

Agent's desired state , $\rho_i(t_f) = \{x(t_f) \quad \dot{x}(t_f) \quad \ddot{x}(t_f)\}$

Goal node range, d_{goal}

Link connection range, d_{link}

Path node similarity range, $d_{similar}$

Similarity ratio threshold, $S_{threshold}$

Output: MA-RRF Forest, $\aleph$

Number of tree branches, τ_i

Number of forest links, l_{RRF}

Multi-agent feasible paths, $\Im_i$

1: $t \leftarrow 0, \Delta t \in [t_0 \quad t_f]$

2: Place safety boundaries around all obstacles

3: Initial number of branches, $k_{branches} \leftarrow 0$

4: Initialize forest, $\aleph$ with roots at $\rho_i(t_0)$

5: **while** $t \neq t_f$

6: **for** each agent i

7: Generate random sample node across free space, η_{sample}

8: Run Tree Expansion **(ALGORITHM II)**

9: **if** added branch, η_{sample} is within the distance of d_{goal} from the agent's goal node $\rho_i(t_f)$

10: Store as $\eta_{i\ goal}$

11: **end**

12: Update forest, $\aleph$ multi-tree links **(ALGORITHM III)**

13: **end**

14: $t \leftarrow t + \Delta t$

15: **end**

16: Run Path Extraction (**ALGORITHM IV)**

17: Run Redundant Path Pruning (**ALGORITHM V)**

Algorithm II shows the connection process between MATLAB's uniformly distributed randomly generated sample node and an agent's tree branch. As priorly discussed, the expansion of nodes for individual tree, τ_i replicates the expansion process for the standard RRT algorithm. A sample node is placed across the environment and is connected to the nearest collision free branch on the agent's tree. This algorithm tests if the link between two nodes is collision free by checking if the node-to-node line intersects with the boundary plane of any obstacle. The parent-child relationship between each tree branch is stored within the shared database under the agent's name. In the MA-RRF planner, this process is repeated for each agent. Thus, four sample points are generated across the test space for each agent. Each agent's tree continues to build branches and expand within the same environment. These parallelly generated trees quickly explore space around them and also advance towards each other through the use of a simple greedy heuristic. These open branches within these individual trees will be used to build the full MA-RRF forest, $\aleph$.

ALGORITHM II Tree Expansion

1: Determine distances between agent's tree branch end node, τ_i and random sample node, η_{sample}

2: Identify tree branch end node, $\tau_{i,neighbour}$ that is closest to the to sample node

3: **if** $\tau_{i,neighbour} \rightarrow \eta_{sample} \notin \xi_{obs}$

4: Add new collision free branch to tree, $\tau_{i\ new}$

5: $k_{branches} \leftarrow k_{branches} + 1$

6: **else** no feasible branch connection exists

7: terminate and rerun main program (**ALGORITHM I**)

8: **end**

The most crucial portion of the MA-RRF algorithm is the creation of the forest. Algorithm III shows the steps that are undertaken to merge the individual trees within the test space. Initially, there is close to zero links within a test space. This is because the trees are not within close range of each other. Connecting trees within extremely constricted environments can be challenging as well. In this case, the algorithm can not find collision free links between the trees. As the algorithm progresses, more sample

nodes have been spread across the environment. This allows the trees to expand and the open branches of these trees are soon within each others range. The open branches are connected to create a connection between the different trees. These links create the backbone of the final forest.

At the initial stages of building the MA-RRF planner, all possible linkages were considered. If two open branches within two different trees are within close range of each other, d_{link} it is flagged as a possible connection. Then, testing is performed to determine if it is a collision free link. If an obstacle free linkage is possible, a new connection between trees is created and stored into the database. There are two issues that arise with this method of linking individual RRT trees.

Figure 3.7(a) shows that the number of links increases exponentially to the number of tree branches that are created across a test space. From experimentation, it is found that the creation of too many links leads to the extraction of paths with high similarity. This is due to the large number of links that connect similar subsections between different trees. Thus, this leads to a final collection of highly identical paths. This shows that many of the links that were created are redundant. The second challenge that arises with this manner of linking trees is that it leads to longer processing time during path extraction. The large number of links creates a huge number of unique but similar paths. The algorithm will have to extract each path individually which increases the total run time of the MA-RRF planner. These paths will contain many similar subsections and be removed from the database during the final filtering process.

Here, linkage is applied where dissimilar links between trees are encouraged. These dissimilar links connect different parts of the MA-RRF forest. This increases the possibility of obtaining a diverse collection of multi-agent paths. It also aims to reduce the processing time that is required during the extraction of unique path subsections. A cost function could be introduced in order to create strategically placed links but this would increase the complexity and processing time of the MA-RRF planner considerably.

On the other hand, a simpler method is implemented as shown in Algorithm III. When a new branch is added to an agent's tree, it is an open branch. The algorithm chooses one neighbouring agent's tree at random. Then, the open branch attempts to create a link with the nearest collision free branch within the other agent's tree. The simulation process in Figure 3.7(b), shows that this method simplistically reduces the number of links within the entire forest. It also requires each tree to attempt a connection with a different tree at every iteration. The links are stored within the shared database with variables such as the two endpoint nodes and the trees that they originate from.

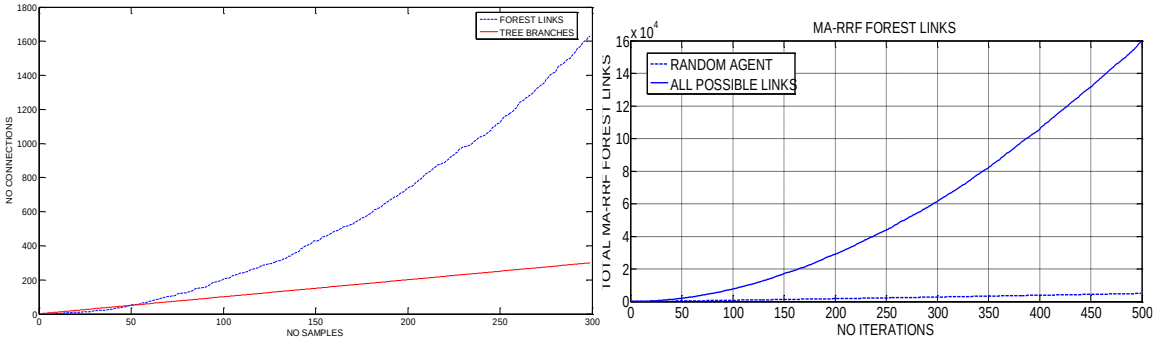


Fig. 3.7. (a) MA-RRF's tree branch and forest links (b) Links of all agents vs. one random agent's tree branch.

ALGORITHM III Forest creation

1: Merge trees generated by each agent into forest, $\aleph = \left\{ \sum_{n=N_A}^{1} \tau_n \right\}$

2: Randomly choose a neighbouring agent, $j \neq i$

3: **for** chosen tree, τ_j

4: Determine if the endpoint of current agent's new open branch, η_{sample} is within the range, d_{link} of other agent's tree branches.

5: **if** open tree branches are within close range of each other

6: **if** connection is collision free, $\tau_j \rightarrow \eta_{sample} \notin \xi_{obs}$

7: Add link between trees, $l_{RRF} \notin \xi_{obs}$ into multi-agent shared database

8: **end**

9: **end**

10: **end**

11: Update forest, $\aleph$

After testing for possible links between trees, the algorithm tests if the newly added sample node, η_{sample} is within close range, d_{goal} of the goal nodes. As previously defined, each agent has its own destination node, $\rho_i(t_f)$. The endpoint of a forest branch, η_{goal} that is close to an agent's goal node can be a part of its own or any neighbouring agent's tree. This process of finding more η_{goal} is repeated at every iteration until the free space mapping process is completed. After the MA-RRF forest is full generated, the algorithm outputs a set of nodes that close in distance to the goal nodes of each agent.

In the first case, both the tree branch, $\eta_{goal} \subset \tau_i$ and the close-range goal node, $\rho_i(t_f)$ are from the same agent, i. The extraction of the path from initial node to the goal node is performed through Algorithm IV with the identification of intermediate nodes. This process is similar to the basic RRT path extraction process. In the second case, the tree branch, $\eta_{goal} \subset \tau_{j \neq i}$ originates from one agent whereas the goal node is for another agent, i. The complexity of extracting a unique path across multiple trees is

higher. The previously stored linkages are paramount to successful extraction of a high number of unique paths. The various connections between the two trees can be obtained by extracting the forest linkages between those two trees. Each link connects a different set of nodes between the two trees. Thus, each unique link creates another unique collision free path. Each unique path is a mesh of three subsections. The first section holds the nodes from the agent's initial point to the start of the tree-to-tree link. The second subsection contains the nodes within the tree-to-tree link. Lastly, the endpoint of the tree-to-tree link to the goal node is extracted to complete the path.

ALGORITHM IV Path Extraction

1: **for** each agent, i
2: **for** each branch within 50*m* of the agent's goal node, $\eta_{i\ goal}$
3: **if** branch near goal node is within agent's tree, $\eta_{i\ goal} \subset \tau_i$
4: Identify feasible paths intermediate nodes $\rho_i(t) \rightarrow \rho_i(t+\Delta t) \notin \xi_{obs}$
5: **else** branch near goal node is within another agent's tree, $\eta_{i\ goal} \subset \tau_{j \neq i}$
6: Extract possible linkages, l_{RRF} between trees τ_i and τ_j within forest.
7: Identify feasible path's intermediate nodes $\rho_i(t) \rightarrow \rho_j(t+\Delta t) \notin \xi_{obs}$
8: **end**
9: **end**
10: **end**

The end user has the option of generating paths that are a mesh of all four trees by extracting all the possible links between all trees. This process will produce a larger number of unique paths but it will require a longer processing time. In this study, each path is a combination of two trees only. At this stage, each agent has tens of thousands of unique feasible paths. These paths will be applied as the GA's initial population within the optimization process. This initial population must be filled with diverse paths. This work defines path diversity through the direction that the path takes across each environment. An initial path population that is diverse will produce a variety of new paths across each generation. Thus, the final stage of the MA-RRF process is the derivation of non-similar paths.

Algorithm V defines the MA-RRF's path pruning process. The unique paths that have been extracted for an agent are compared to one another. This process aims to filter out paths that are too similar in direction. Firstly, the node-to-node distance, $d_{k,l}$ between two paths is calculated. Then, if two nodes are within close range of each other, $d_{similar}$ it is considered as a similar node. In this study, $d_{similar} \leq 10m$. Each path is then assigned many similar nodes, N_s. Next, the similarity ratio, S for each path is

determined by comparing the number of similar nodes with its total nodes. If the number of similar nodes exceeds the similarity threshold, $S_{threshold}$ it is labelled as non-unique and removed from the agent's path database. Each path is compared to all the other paths to assure that there aren't two highly similar paths within the agent's initial path population. This filtration process lays the foundation for trajectory diversity within the next stages of the algorithm.

ALGORITHM V Redundant Path Pruning

for each agent, i
  for each feasible path, $\Im_i$
  Initial number of similar nodes, $N_s \leftarrow 0$
    for each path node, $\eta_i \subset \Im_i$
    Determine distance, $d_{i,j}$ between current node η_i and other path nodes, η_j, $j \neq i$.
      if $d_{i,j} \leq d_{similar}$
      Number of similar nodes in current path, $N_s = N_s + 1$
      end
    end
    Calculate similarity ratio, $S = N_s / N_{total}$
    if similarity ratio, $S \geq S_{threshold}$
    Remove path from database
    end
  end
end

3.2.3 MA-RRF INITIAL PATH POPULATION

The MA-RRF path planner provides the end user with a large database of information. The output data from the planning and free space mapping process is easily viewable through high resolution imagery. The MA-RRF forest progression is shown across all three test environment. Figure 3.8, 3.10 and 3.12 show the trees that are created by four agents across each test environment. Here, each agent's tree branch is defined by its different colours. The MA-RRF forest links are shown in black. These figures highlight the challenges that are present within these different test spaces. Similarly, the final path collection in Figure 3.9, 3.11 and 3.13 also show the end user the possible trajectories that each quadrotor will track during real life flights. It can be seen that the paths are well spread across the environments and are diverse in direction.

The end user is also presented with knowledge regarding the number of unique collision free paths that were initially extracted. Table 3.2 shows the number of iterations and linkages that are present within each test environment. The number of iterations that

is required for each environment is dependent on the minimal number of paths that can be extracted after the path pruning process. The initial population for the next stage GA will require a collection of diverse paths in order to create the new generation paths. Here, the impact of using different similarity thresholds when filtering similar paths is shown as well. More than triple digit paths are expected for 80% similarity threshold whereas there are double digit paths with less than 65% similar nodes. Lastly, a minimum of single digit number of paths is expected when 50% similarity threshold is applied. The minimum number of paths allows the user to make an easy comparison between similarity percentages versus number of paths for all the test spaces. It helps the end user to determine the level of diversity that is required within the final path population.

The first test space is the high-rise cityscape environment. There are three challenges that presented itself during the planning process. Firstly, the size of the boundaries around the buildings must create a balance between forest progression and the avoidance of building edges. The sharp edges must be safely avoided by all agents. Another issue with the cityscape environment is progression of the MA-RRF forest isn't as quick as the other environments. The surrounding safety zones will further restrict the progression of the individual trees since the roads between the buildings are already narrow. Lastly, it can be difficult for the MA-RRF algorithm to attempt to merge two trees together within this test space. Each high-rise building is created to be equal in height. This means that the tree branches from different trees are constantly separated by a building. It can be tough to create tree-to-tree links unless the open branches are within the same narrow road.

Figure 3.8 shows that MA-RRF forest progression across the tall buildings. As visible, the free space is fully explored by the tree branches from the ground to the roof of the buildings despite space constrictions. The MA-RRF forest and its individual trees are bounded to the spaces that are in between the buildings. There are no safety zone breaches. Thus, the branches are able to maintain a good distance from all building edges. The imagery also proves that the individual trees aren’t as well spread as the other test environments. Each agent’s tree is denser in areas that are close to its root node. There are many tree-to-tree linkages across the middle area of the cityscape environment. Table 3.2 shows that the MA-RRF planner is run for 700 iterations within the cityscape test space. It requires more sample points per agent in comparison to the indoor environment and mountainous terrain. The high-rise cityscape environment requires 700 sample nodes per agent in order to generate the similar number of diverse paths. The algorithm is run for a longer period because it allows the individual tress to cross over to the other parts of the environment. This will encourage linking between all trees. The narrow passages that exist between buildings constrict the number of long range links within the MA-RRF

forest. Thus, the cityscape space has the lowest ratio of amount of links over iterations between all three environments.

Next, the path extraction and pruning process produces the final paths that create the initial GA path population. Table 3.2 shows that the cityscape environment has the lowest ratio of unique paths per iteration. This is due to the low number of links between the individual trees. It is also caused by the low number of forest branches that are close to the goal nodes of all agents. Another pattern that emerges from the data is the relationship between the maximum number of unique and diverse paths. Agent 3 has the highest number of near-to-goal nodes and unique paths. It also has the most number of non-similar paths after the path pruning process. Figure 3.9 shows the paths designed by the MA-RRF algorithm for four agents within the high-rise cityscape environment. The paths with less than 65% similarity are applied within the multi-agent quadrotor trajectory optimization algorithm. Here, each colour defines one path. With all four agents, it can be seen that a diverse set of paths have been designed by the path planner. These paths are well spread by width and height of the free space narrow passages between the buildings. Some paths have nodes across lower ground level whereas others are placed are higher heights. The MA-RRF path planning algorithm has successfully designed paths that do not have any collision points with the surrounding buildings.

The second test environment is the highly cluttered indoor space. Similar to the cityscape environment, this test space has its own set of challenges. Firstly, the MA-RRF forest must be capable of detecting the small number of entry points across the indoor space. These entrances are small in comparison to the size of the walls that hold them. The safety zones that are added around each entry way adds further complexity. A constant issue during early experimentation is the definition of safety zones around the smaller windows. A balance in boundary size must be struck between being too large or small in range. The small entry points become even smaller with the addition of large sized buffer regions. Thus, it can be highly challenging for the free space mapping algorithm to detect it. The forest branches cannot penetrate the large boundary and cross over to the next room. The application of smaller radius boundaries instead can cause the branches to collide with the sharp corners of the windows. Secondly, the varying sized clutter within each room must be adequately mapped by the MA-RRF forest. These obstacles vary in size and are floating in three dimensional spaces unlike the cityscape environment. The branches must progress over and under the many clutter cubes. Lastly, creating forest links that bridge trees in different rooms can be difficult. Only a few tree branches are able to make a connection between two nodes that are within different rooms. Similarly, it can be hard to create tree-to-tree links that bridge two rooms. The forest links must also be capable of making multi-tree connections within each room.

TABLE 3.2. MA-RRF EXPERIMENTAL RESULTS.

Environment	Iterations (run time)	No Forest Links (colour)	No Agent (colour)	Initial Position (m)	Target Position (m)	No Near Goal Nodes	Unique Paths	S_1	Filtered Paths	S_2	Filtered Paths	S_3 (run time)	Filtered Paths
CITYSCAPE	700 (0.084s)	2443 (black)	1 (green)	[60,220,10]	[230,100,190]	18	7578	80%	105	65%	22	50% (1014.74s)	4
			2 (magenta)	[220,200,5]	[20,40,195]	11	2047		65		18		4
			3 (blue)	[200,10,10]	[25,180,185]	29	7799		110		26		8
			4 (red)	[40,75,5]	[200,230,195]	13	2489		44		10		5
HIGHLY CLUTTERED INDOOR	750 (0.159s)	5778 (black)	1 (green)	[80,105,80]	[20,-5,20]	50	37399	80%	121	65%	18	50% (325.79s)	6
			2 (magenta)	[98,-5,50]	[10,105,30]	85	65761		193		18		5
			3 (blue)	[10, -5, 90]	[90,105,10]	51	43076		116		15		8
			4 (red)	[20,105,70]	[90, -5, 10]	47	39281		174		34		6
MOUNTAIN TERRAIN	500 (0.047s)	5758 (black)	1 (green)	[9.74,9.85,36.74]	[60.89,55.04,46.49]	79	58007	80%	735	65%	75	50% (1692.14s)	17
			2 (magenta)	[62.27,9.82,27.22]	[6.15,55.19,47.50]	208	151052		1290		60		8
			3 (blue)	[4.75,55.05,44.63]	[61.30,9.67,47.87]	678	494242		3693		135		15
			4 (red)	[59.78,62.46,47.84]	[7.63,9.73,35.11]	762	528869		3328		88		12

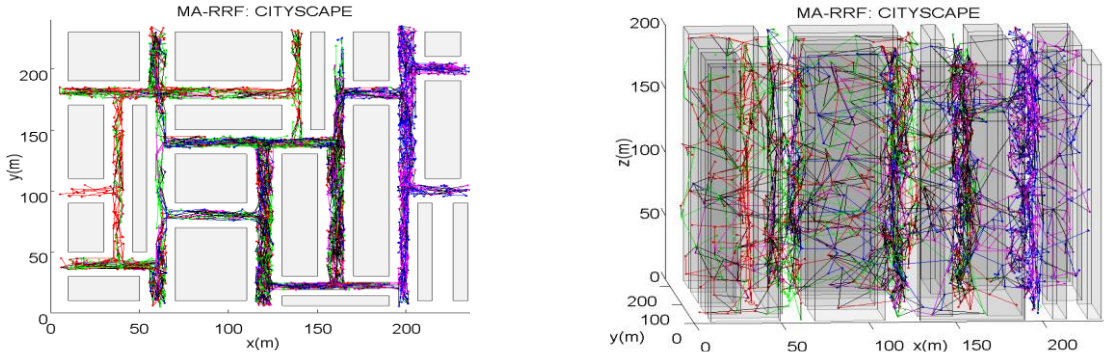
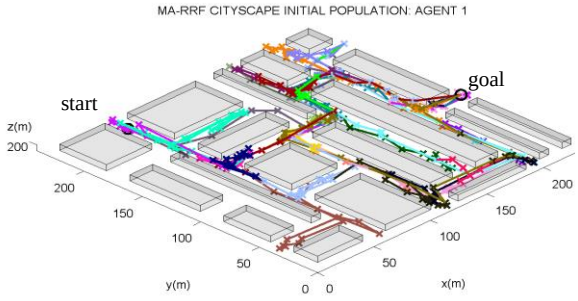
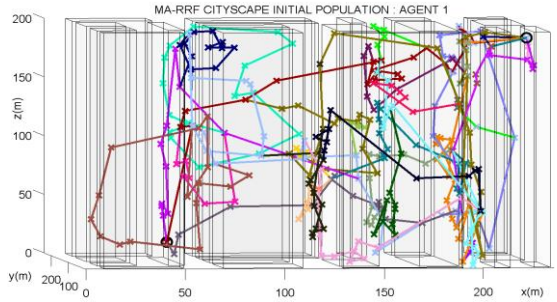


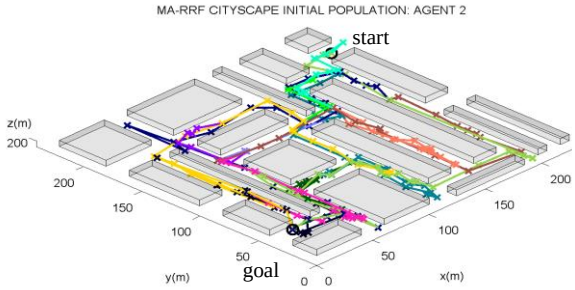
Fig. 3.8. MA-RRF full space exploration across the cityscape environment.



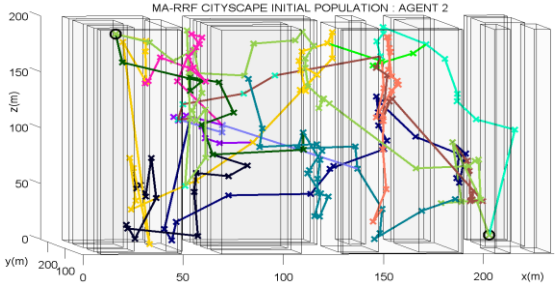
(a) Top view of agent 1's paths.



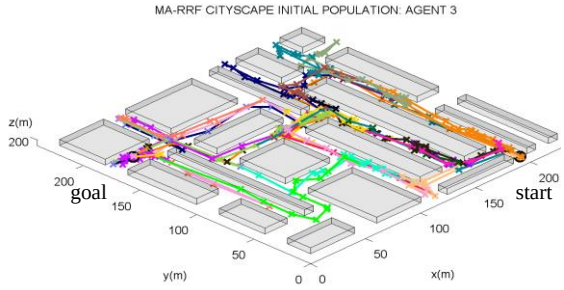
(b) Side view of agent 1's paths.



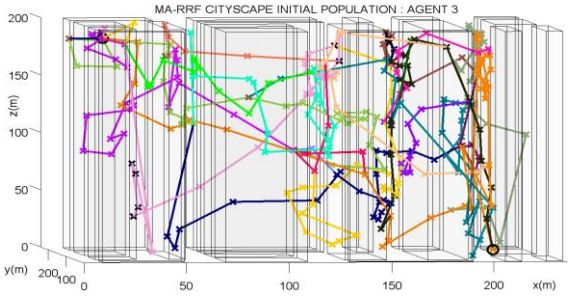
(c) Top view of agent 2's paths.



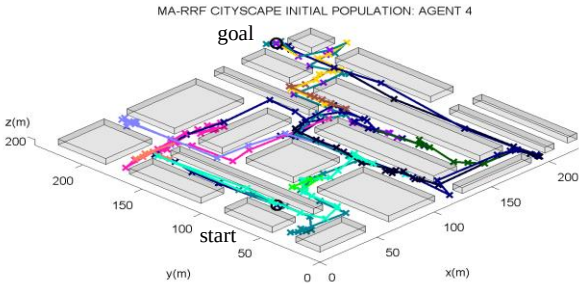
(d) Side view of agent 2's paths.



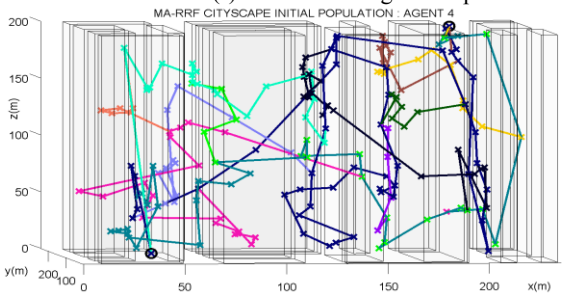
(e) Top view of agent 3's paths.



(f) Side view of agent 3's paths.



(g) Top view of agent 4's paths.



(h) Side view of agent 4's paths.

Fig. 3.9. MA-RRF diverse paths extracted across the cityscape environment.

Figure 3.10 shows the forest progression across the highly cluttered indoor space. Similar to the cityscape environment, each agent's tree is shown in a different colour. The MA-RRF forest links are shown in black. Initially, the size of the obstacles safety zone is estimated through multiple test simulations. This boundary size creates a balance between MA-RRF forest progression across each room and collision avoidance. It can be seen that the MA-RRF branches are capable of exploring different rooms through various entry points. The algorithm is capable of identifying all entry points despite the lack of windows and doors. This allows a few branches to transition between rooms and creates nodes across the entire environment. Thus, full free space exploration is successfully performed from ground to ceiling of the environment. Table 3.2 shows that the highly cluttered indoor space proves to be the most challenging environment. Here, 750 sample points per agent is necessary to extract a high number of diverse paths from the environment. This is because of the minimal number of entry points that are placed across each wall. On the other hand, a large amount of forest links is created due to the long run time as well as the availability of free space inside each room.

The branches of the MA-RRF forest create the GA's initial path population. Table 3.2 shows that the number of filtered paths based on similarity is a very small percentage of the unique paths. This occurs despite the high number of unique paths that were extracted. The indoor space extracts almost ten times the number of unique paths in comparison to the cityscape test space. It also has a higher number of near-to-goal nodes. The reason for the lack of diverse paths is due to the constraints of the small number of entry points. This direction that paths can take is limited to just these windows and doors. Also, the large sized obstacles within a room force these paths to avoid it through similar translational and angular motions. Agent 2 has the most number of unique paths but it doesn't have the highest number of diverse paths. This means that the paths that were designed for agent 2 are many but are similar in direction. The diverse collection of paths for all four quadrotor agents is shown in Figure 3.11. Based on prior experimentation, paths with less than 65% node similarity will progress to the optimization stage. This decision is made to encourage the recombination, mutation and maintenance of diverse paths during the initial generations. The four agents are able to progress across different entry points and reach every room. The path nodes for all agents are well spread. The most important criterion is the planner's ability to avoid any collisions. The paths of each agent flow over and under the clutter in order to avoid any possible collisions.

The last test environment is the mountainous terrain. It has a larger amount of free space in comparison to both the other environments. Still it has many small sized peaks and some large peaks across the terrain. It can be difficult to set a constant start and goal node for the agents because the peaks are always in a different location within different simulations of the terrain. Therefore, the start and goal nodes for each agent is randomly generated across the four corners of the terrain. It is accepted as the initial and goal nodes

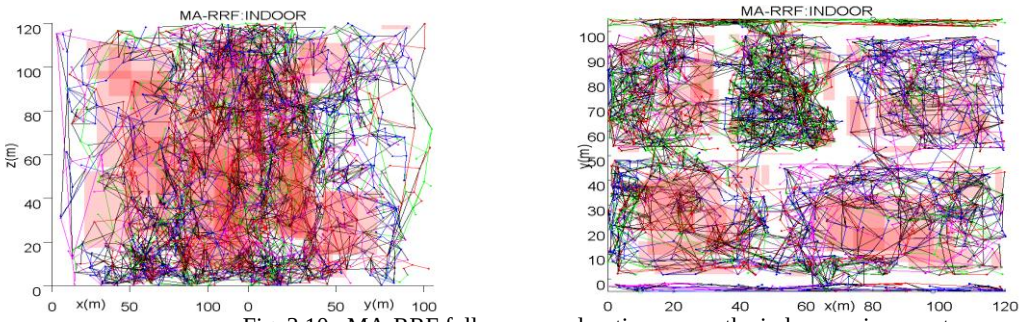
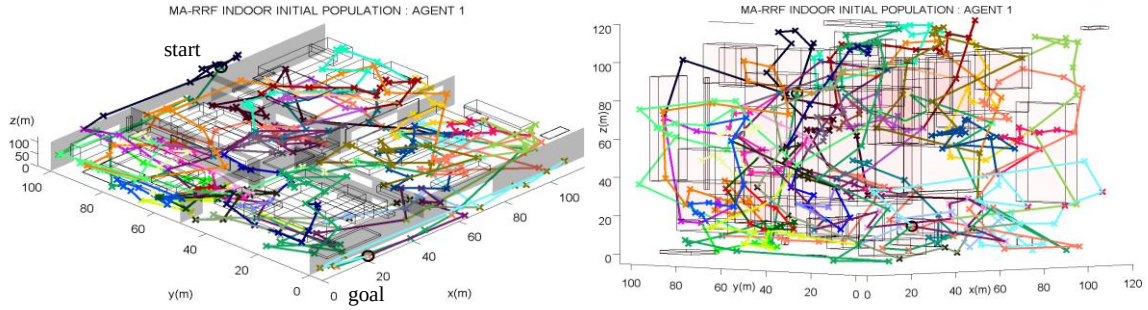
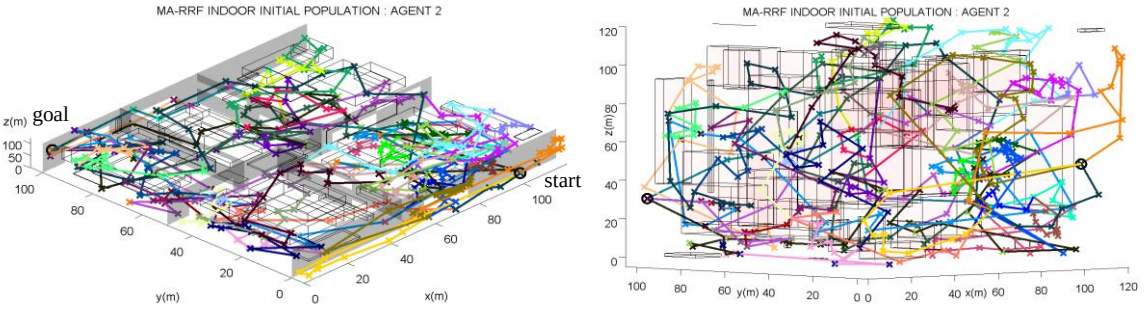


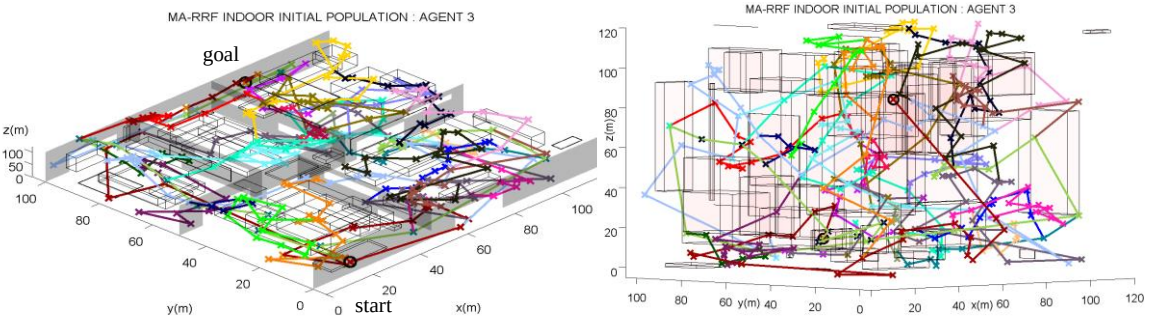
Fig. 3.10. MA-RRF full space exploration across the indoor environment.



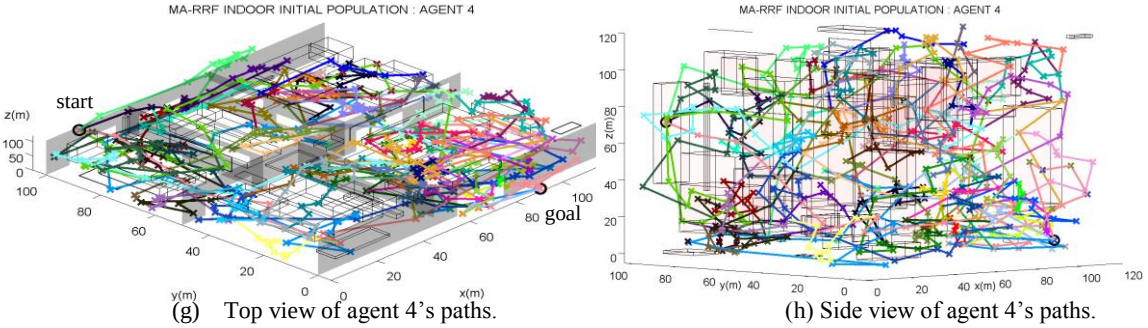
(a) Top view of agent 1's paths. (b) Side view of agent 1's paths.



(c) Top view of agent 2's paths. (d) Side view of agent 2's paths.



(e) Top view of agent 3's paths. (f) Side view of agent 3's paths.



(g) Top view of agent 4's paths. (h) Side view of agent 4's paths.

Fig. 3.11. MA-RRF diverse paths extracted across the indoor environment.

if it doesn't collide with any mountain peaks. An example of these random nodes with decimal values are shown in Table 3.2. In this test space, the MA-RRF forest must be able to avoid all the various sized peaks. During many early simulations, the branches could avoid areas with larger ripples across peaks. However, the branches tend to collide with locations that contain smaller changes in height. These are areas where the path nodes would slightly graze the corners of these smaller peaks. This is because the safety boundaries are positioned through an approximation of terrain heights. It is based on the peak height at all four corners of each grid. This produces an angled diagonal top plane. In this case, the other ripples within the grid are not represented. An improvement was made by positioning the safety zone based on the maximum height of all minor peaks within a grid. Now, all minor peaks within each grid are accurately represented. This method reduces processing time and produces collision free paths. Thus, a full collision free forest is created by MA-RRF algorithm for the mountainous terrain.

Figure 3.12 shows the progression of the forest across the mountainous terrain. The branch of each tree is defined with a different colour as with the other environments. The linkages between different trees are shown in black. As predicted, the MA-RRF forest easily progresses across the terrain. The imagery shows that the forest can explore between high peaks and lower depths within the mountainous terrain. Many paths tend to exist at higher heights which have larger amounts of free space if the local minima aren't well explored. The MA-RRF forest is denser across the high peaks of the mountain. This is due to the large amount of free space that is present between the mountain peaks. On the other hand, it is sparse at areas that are closer to the ground level. The bases of the mountains have narrow passages between them. This constricts the progression of the tree branches across the lower parts of the terrain. Still, the planner has been able to successfully map the free space across the entire terrain. Table 3.2 shows that the mountainous terrain easily extracts the minimum number of paths. These paths are designed within 500 sample points per agent. This test space requires the lowest sampling rate between all environments. It also manages to create the highest number of forest links. This is because it is the least constrained environment amongst all the test spaces.

Next, the unique paths for the team of four quadrotors are extracted from the MA-RRF forest. The average numbers paths per agent post the exploration and filtration process is the highest. This is because the mountainous terrain has the most number of near-to-goal nodes as well. Many forest branches have been able to reach the large amount of free space around each goal node. Still, the number of unique paths for each agent differs greatly unlike the other environments. Some goal nodes are more easily reachable than the others due to the variation in height. It also depends on the number of peaks that surround the goal node. Here, the paths with less than 50% similarity are

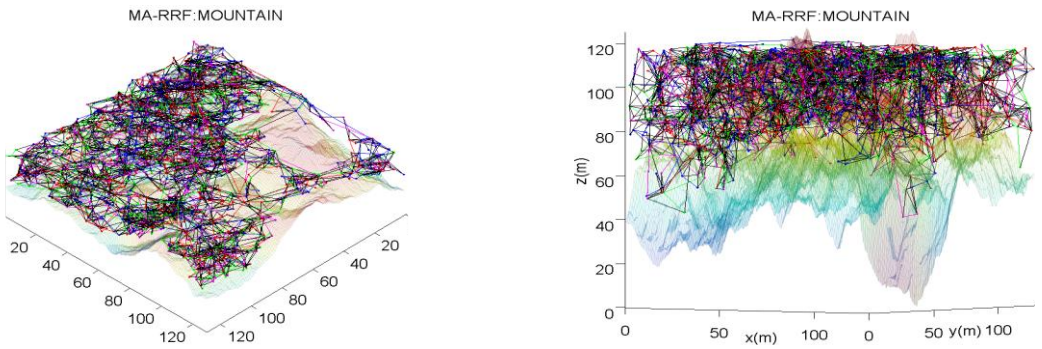
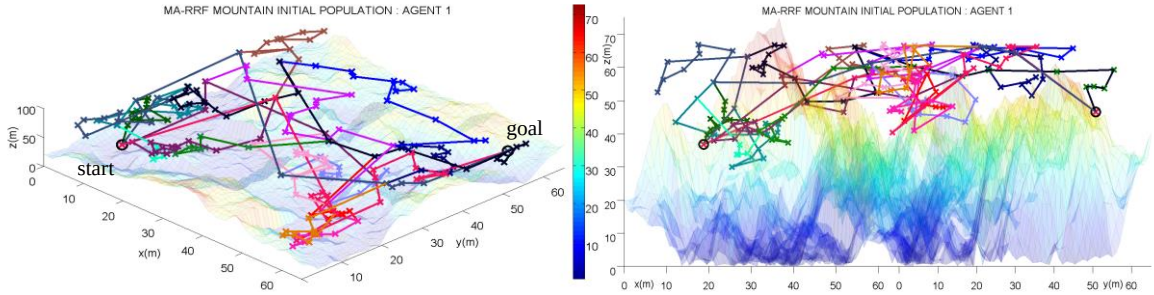
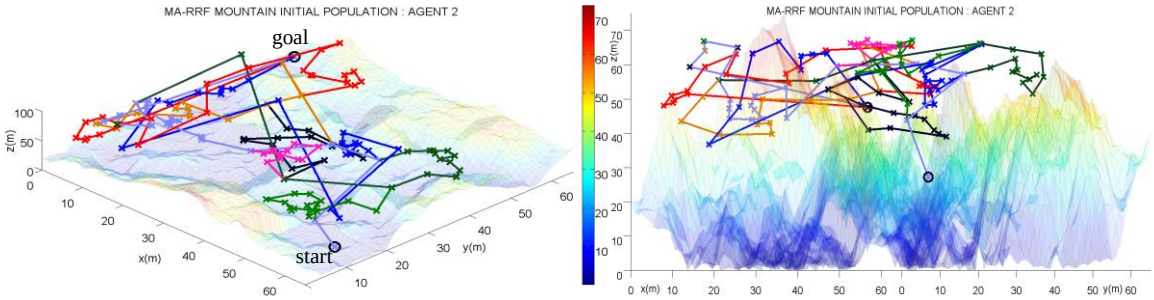


Fig. 3.12. MA-RRF full space exploration across the mountainous terrain.



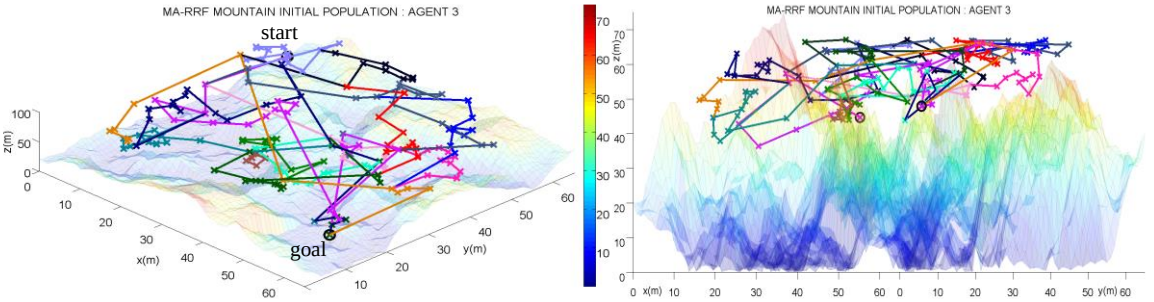
(a) Top view of agent 1's paths.

(b) Side view of agent 1's paths.



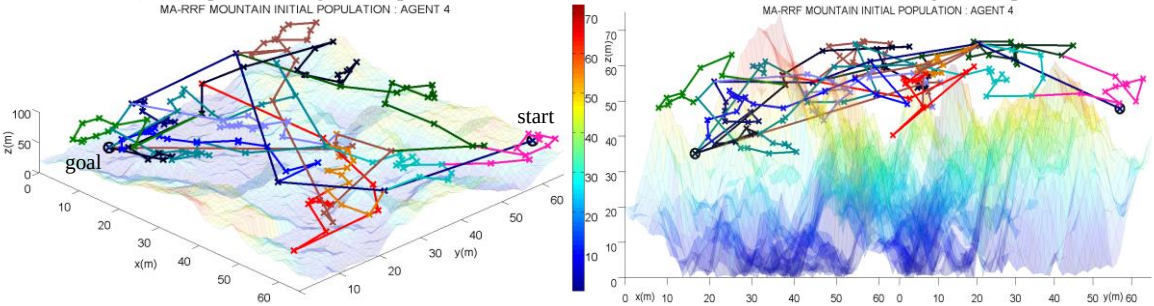
(c) Top view of agent 2's paths.

(d) Side view of agent 2's paths.



(e) Top view of agent 3's paths.

(f) Side view of agent 3's paths.



(g) Top view of agent 4's paths.

(h) Side view of agent 4's paths.

Fig. 3.13. MA-RRF diverse paths extracted across the mountainous terrain.

maintained within the initial population. The various paths for each quadrotor are shown in Figure 3.13. The planner has designed collision free paths with waypoints within an environment with different levels of elevations. It is shown that the paths for all four agents have nodes at various heights. It can be seen that neither the mountain peaks nor local minima are neglected. The diversity in terms of grid exploration is less than the prior two environments. There is a decrease in paths that progress across the valleys between high peaks that are close in distance. Thus, a higher percentage is applied for the path filtration process.

These diverse and collision free paths will be applied as the building blocks for the next stages. It will be possible to maintain a good spread of solutions with various trade-offs across many generations if the initial MA-RRF paths are truly diverse. As previously defined, these paths will be hybridized by GA to form the initial population for the many-objectives optimization process.

3.3. GENETIC ALGORITHM

Evolutionary algorithms (EA) are inspired by the concept of 'survival of the fittest' within a population. This concept is derived from the parent-child genetic relationship that exists within the human population, wildlife and nature. Two parents that contain certain genetic characteristics will often produce an offspring. This new offspring will have a mix of both the parent's genetic material. These genes are made from a combination of dominant and recessive characteristics. Thus, the offspring will appear to have the traits of the dominant gene. In many cases, the offspring of two parents aren't exactly similar to their ancestors. This is because external factors such as nutrition or environmental changes can cause a mutation in the genetic makeup of an individual. The genetic materials and their variations describe process of evolution. The variation that exists within a new generation allows it to survive for a longer time than its ancestors. The same variations can also cause it to live a shorter lifespan. The process of evolution slowly removes genetic material that can be harmful and maintains characteristics that promote a longer lifespan. Humans often pick partners that appear to be healthy. Thus, each new generation contains children with good health as well.

The population of any species continues to promote the creation of healthy individuals. In this case, the objective of evolution is to preserve a species and avoid extinction. The cost functions that describe the lifespan of an individual can be defined through their ancestors, diet, lifestyle and environment. Individuals with the best cost function values are maintained within its species. These concepts can be applied within an optimization problem. Most optimization problems require an algorithm that can quickly find the best solutions within a search space. It needs an algorithm that can effectively detect the minimum point within complex cost surfaces. It must also be

capable of converging towards search areas that contain solutions with good characteristics. This creates a final population that is filled with good solutions. Then, the end user will have the option of picking the best solution for their optimization problem.

The most commonly utilized EA is genetic algorithm (GA). The basic process of GA is shown in Figure 3.14. GA is ideal for the generation of well minimized and diverse trajectories for multi-agent quadrotors. This is due to its large population size per iteration. It allows the end user to generate a large collection of paths for multiple agents simultaneously. It is also suitable for solving many-objectives problems. The selection process within GA can be assisted by the many-objectives optimization methods. Here, the ranking of solutions at each generation can be performed by a many-objectives optimization (MOO) algorithm. These trajectories will be ranked by MOO based on the objective values of the many objectives functions that are considered within GA. The MOO algorithm promotes the maintenance of trajectories that are well minimized and diverse across generations. The application of elitism within the selection process will also allow the end user to keep well minimized trajectories across generations. These trajectories will remain untouched throughout the optimization process. GA is suitable for a multi-agent system because it is able to run on parallel computers. In this case, it can be further sped up because GAs can be broken down to as many parallel computations of the solutions per generation simultaneously. The level of parallelism is only dependent on computing power thus easily expandable to fit swarms of agents.

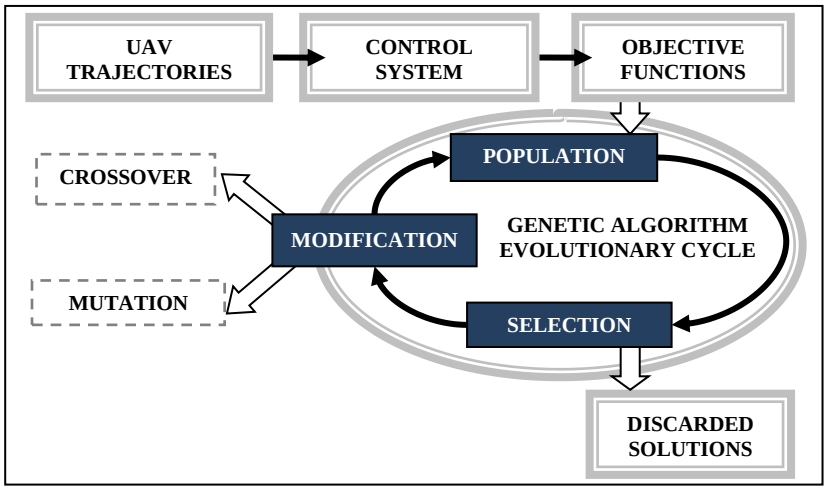


Fig. 3.14. Genetic algorithm evolutionary cycle.

The understanding of biological reproduction mechanisms is necessary for the application of GA. The various steps within the GA are natural selection, mutation and genetic recombination. The parent selection process can be implemented through top to bottom pairing, tournament selection and random pairing. The most simplistic method would be the top to bottom pairing of solutions. Here, selection starts at the top of the population

pairing odd and even rows together. Another highly utilized selection method is randomized pairing. This process requires no sorting and requires minimal processing time. There are two types of this pairing which are the unweighted and weighted version. A popular weighted random parent pairing is the roulette wheel weighting. Tournament selection is equally as popular where random members of the population are chosen to compete through comparisons with one another. This study applies a 50% selection rate as shown in Table 3.3. Thus, half of the previous population is maintained within the next generation's parent population. Figure 3.15 shows how GA is applied within this research. It can be seen that the simplistic unweighted random pairing is used for the parent selection process.

After the selection of two parents, two new child solutions are created from them. The next generation is built through the GA's crossover and mutation process. The crossover process picks apart the parents into subsections and merges these subsections to create new offspring. Many variations of the crossover operator exist such as the single-point crossover, partially matched crossover (PMX), order crossover (OX) and cycle crossover (CX). Flowchart 3.15 shows that the single-point crossover is utilized in this study. Next, the mutation operator introduces an element of randomness within the offspring. This process diversifies the new solutions from their parents. It also allows the algorithm to explore new search areas. There can be negative consequences if the mutation probability is too high. Large changes within an offspring can destroy a good solution. Thus, it is important that the algorithm creates minor changes within an offspring. There are many variations of the mutation operator. Popular versions are the external, internal, and omission mutation operator. It can be seen in Flowchart 3.15 that this work applies all three versions.

The new offspring are often filtered based on their feasibility and level of similarities. Many works remove redundant information from their solution population. Redundant data can be introduced within an offspring during the crossover and mutation process. The removal of unnecessary information allows the algorithm to run more efficiently. Similarly, feasibility check is often performed to determine if the child solution will be a viable candidate for the next generation. This process is highly dependent on the application of the GA. Another important post processing option is to maintain a child population that is dissimilar from the parent population. Here, the percentage of uniqueness of the new offspring can be tested. As shown in Figure 3.15, both of these post processing options are utilized within this multi-agent trajectory generation and optimization algorithm. Lastly, the final population of parent and child solutions is created. This population will be carried forward to the next generation's selection process. The algorithm is run continuously until a termination point is met.

3.3.1. PATH PLANNING WITH GENETIC ALGORITHM

The initial population for GA is comprised of the paths that were created by the MA-RRF sampling based planner. There are many studies that choose to apply GA as a standalone algorithm for path planning. The algorithm is suitable for applications that have a small number of agents or a narrow search area. This research performs path planning for many agents within highly complex three-dimensional environments. GA will require a long running time to fully search the entire test space. In this research, the search space is initially mapped by MA-RRF. It quickly identifies the obstacle free regions and generates feasible path nodes for all agents. This process reduces the amount of free space that needs to be searched by GA. It also allows GA to execute more refined multi-agent path planning. Here, GA is not beginning with an arbitrary set of path nodes. The initial MA-RRF path nodes will serve as an indicator as to the distance and possible directions to move towards. This research aims to use GA to create and move the initial path nodes towards more optimal areas within the search space. The application of GA towards trajectory generation as is shown in Figure 3.14 requires additional modifications such as the addition of a path repair mechanism. These changes are defined within the flowchart in Figure 3.15. The variables that are applied along with the GA path planner are presented in Table 3.3.

Any application GA requires a set of input variables. These input variables will be manipulated in order to create a desired output. In this work, the input variables are the initial MA-RRF paths that are obstacle free and diverse in direction. Then, the offspring paths that are designed by the GA will be applied as the input for future generations. The optimization algorithm will test to see if the current input creates the desired output. Here, the output is defined as the values of the many objectives that are used within the MA-Spread and MA-Formation application. The value of each objective function will be estimated by a multi-agent quadrotor control system. Lastly, the multi-agent trajectories will be ranked based on their predicted objective values. GA requires the assistance of a many-objectives optimization algorithm in order to complete the path selection process. MOO will be used to define the level of diversity and optimality of each path. Thus, GA will be able to identify paths that are worth maintaining within the next generation's population. Trajectories that are highly ranked by the MOO algorithm will be used as the parent paths for the new offspring.

3.3.2. PATH POPULATION & NATURAL SELECTION

The selection process for parent population is based on this study's dual application. This work aims to optimize the trajectories of N_A quadrotors for spread and formation flights. For MA-Spread, the best path combination, $C(t)$ for all agents defines the paths

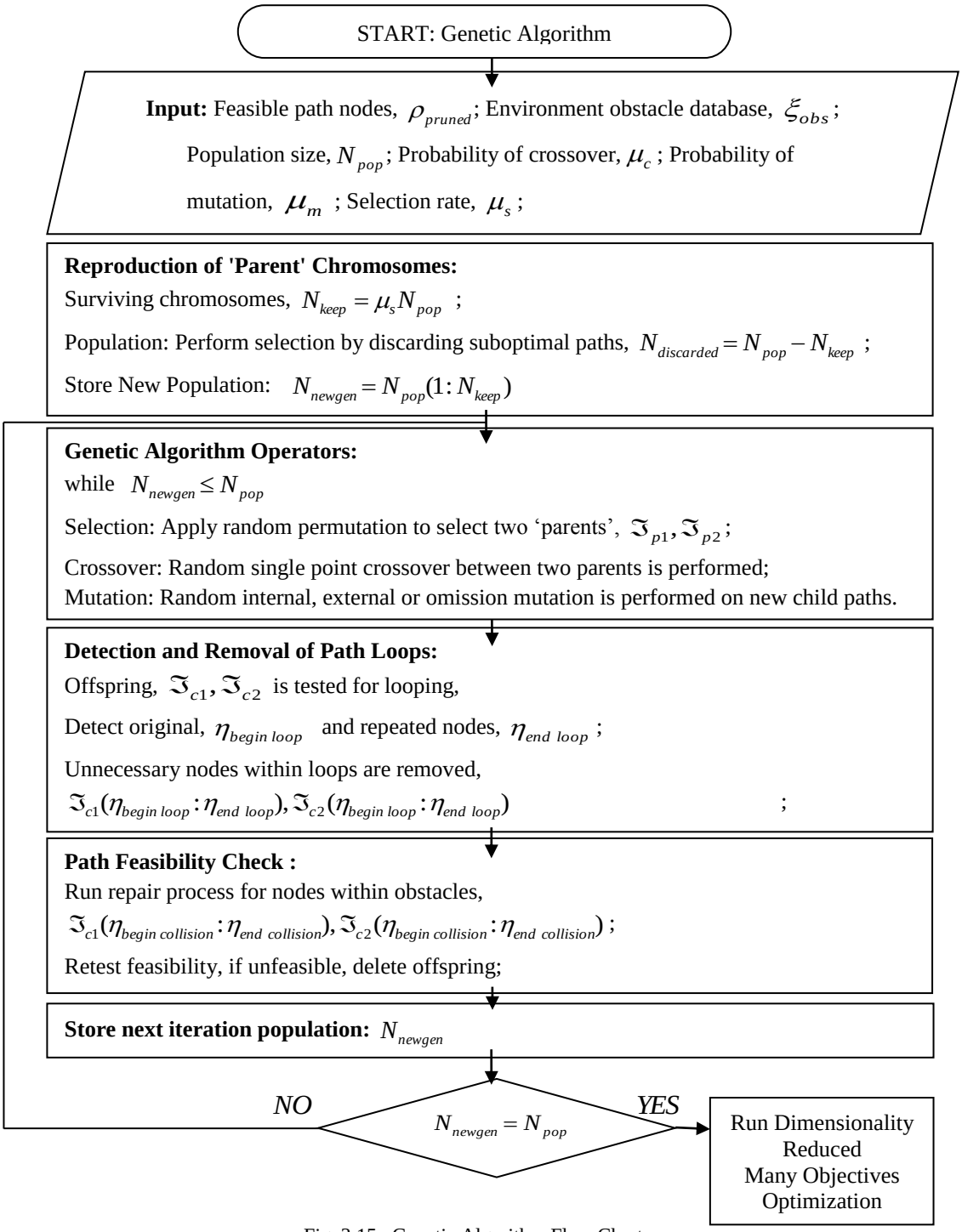


Fig. 3.15. Genetic Algorithm Flow Chart.

that are maintained for each agent. The team of quadrotor's collective objective values will be used as the selection criterion for the parent population. During each selection process, the best path combinations are maintained within the next generation's population. The trajectories, $\Im_i(t), i = 1,.....\mathrm{N}_A$ for all agents are stored within the database based on its path combination. There will be repetitions of individual agent trajectories within different combinations. Thus, the number of unique trajectories per agent varies depending on the survival of the combination that holds it.

The second application is MA-Formation where the population contains the best formation reference trajectories, $\Im_{formation}(t)$. This reference path is needed to define the formation structure of the entire team of quadrotors. These formation shapes will define the trajectories of each individual agent. The objective functions in the MA-Formation mission are dependent on the formation shapes and flight paths of each agent. Here, the formation reference trajectories that produce minimal objective values are maintained within the next generation. In comparison to MA-Spread, all of the formation reference trajectories within a population are unique. The GA that is applied within this research is standardized to both applications. The only variable that requires changing is the number of agents. This study applies a 50% selection rate as shown in Table 3.3.

$$N_{pop} = \begin{cases} C\ (t) = \{\Im_1(t) \quad \Im_2(t) \quad \dots \quad \Im_A(t)\} & for \quad MA-SPREAD \\ \Im_{formation}(t) & for \quad MA-FORMATION \end{cases} \tag{3.1}$$

where t = path nodes.

TABLE 3.3. GENETIC ALGORITHM CONSTANT PARAMETERS

Parameter	Description	Value	Parameter	Description	Value
N_{pop}	Population Size	30	$\mu_{internal}$	Internal Mutation Percentage	0.33
μ_s	Selection Rate	0.50	$\mu_{omission}$	Omission Mutation Percentage	0.33
$\mu_{external}$	External Mutation Percentage	0.33	N_c	Number of crossover points per path	1
$r_{external}$	External Mutation Radius	50 m	N_m	Number of mutation nodes per path	1

3.3.3. PARENT SELECTION

The two parent paths are picked from the pool of trajectories of each agent. The first constraint for next generation trajectory is that both chosen parent trajectories, $\Im_{p1}(t)$, $\Im_{p2}(t)$ must be from the same agent's database. These paths must hold identical initial and goal nodes as visible in Figure 3.16. This facilitates in the construction of new paths that are tailored to the start and final nodes that were initially defined for each agent.

$$\Im_{p1}(t_0) = \Im_{p2}(t_0) \tag{3.2}$$

$$\Im_{p1}(t_f) = \Im_{p2}(t_f) \tag{3.3}$$

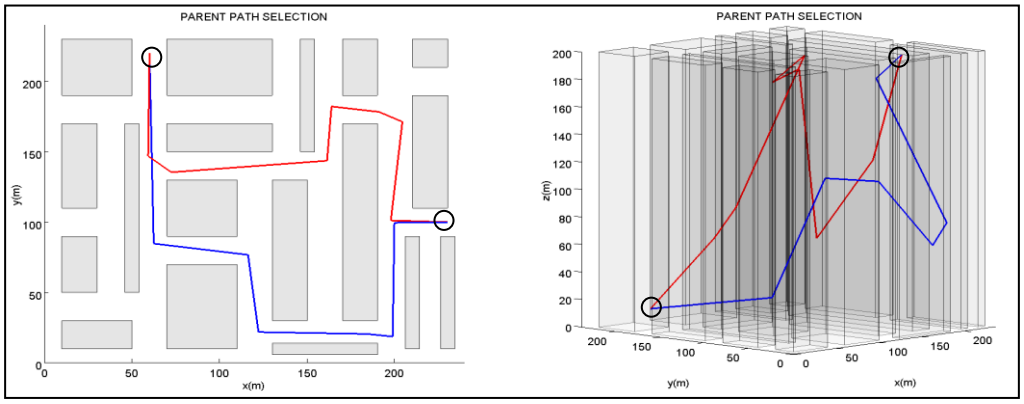


Fig. 3.16. Genetic Algorithm Selected Parents.

The path diversity amongst the parent population is maintained through a filtration process. This is based on user defined thresholds. Prior experimentation shows that the parent population will continue to converge towards similar directions without additional diversity management. This is especially true for multi-agent flights within extremely constricted spaces. In this study, the initial population prioritises higher levels of diversity whereas later populations have a less strict diversity threshold. As previously discussed, a similarity filter is applied with the trajectories created by the MA-RRF stage where the threshold is set to less than 65% between nodes of two trajectories. Thus, this facilitates a diverse initial parent population. As the algorithm proceeds, the filtration process is less strict where the threshold is set to no more than 75% similarity. This method maintains well minimized solutions within the parent population despite having lower levels of uniqueness. The random selection of parents will always pair two diverse parents since the population maintains a minimum diversity measurement through generations.

3.3.4. PATH CROSSOVER OPERATOR

The creation of the next generation offspring is dependent upon the mating process of both parents. The child paths are generated by the crossover and mutation operators. Based on previous experimentation, single point crossovers have a higher chance of producing feasible paths. It is also noncomplex. Alternative methods of crossover can increase repetitive path looping within trajectories and require longer periods for path repair. Table 3.3 shows that the number of crossover points for a two parent paths is one node per path. This leads to a single point crossover for both the parent paths.

Single point crossover for path planning can be executed in two ways. Both options are performed considering all nodes within the parent paths except for the initial and goal node. The first method requires the algorithm to identify a common node between both parent paths. Chances of finding a common node are high because the paths are progressing towards the same destination node. The common node is used as the crossover point for both parent paths. The advantage of using a common node is it creates

feasible child paths. Through simulation, it has been identified that this method also has the disadvantage of creating less diverse child paths. The second method randomly selects a node within each parent path. These nodes will be set as the crossover points for each parent. This method produces the opposite effect. It produces highly diverse paths that contain a small number of obstacle collision points. This research chooses to apply the second option because path diversity is extremely important for the optimization process. Randomised selection of crossover points within the selected parent paths, t_{c1}, t_{c2} is performed as shown in Figure 3.17. This process produces offspring paths, $\Im_{c1}(t)$, $\Im_{c2}(t)$,

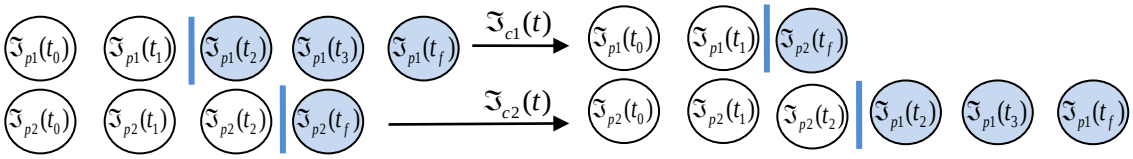


Fig. 3.17. Single point crossover with selected parents.

$$\Im_{c1}(t) = [\Im_{p1}(t_0,...,t_{c1}) \ , \ \Im_{p2}(t_{c2+1},...,t_f)] \tag{3.4}$$

$$\Im_{c2}(t) = [\Im_{p2}(t_0,...,t_{c2}) \ , \ \Im_{p1}(t_{c1+1},...,t_f)] \tag{3.5}$$

As shown in Figure 3.18, single point crossover is performed upon the two selected parent paths. This process results in two offspring that are a mesh of both paths. This method creates offspring that are diverse and have minimal number of collide points.

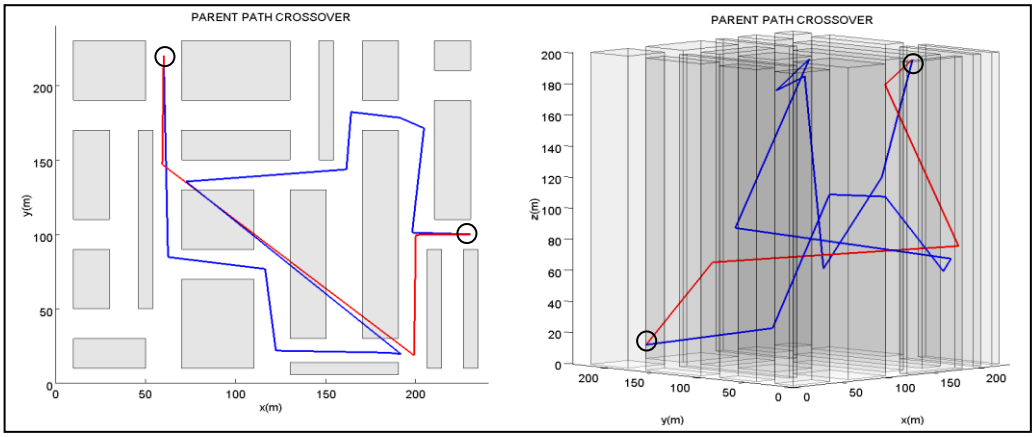


Fig. 3.18. Genetic Algorithm Parent Paths Crossover within Cityscape Environment.

3.3.5. PATH MUTATION OPERATORS

The mutation operator is applied towards both offspring after the completion of the crossover process. Applying the mutation operators can be a constructive or destructive force for the newly generated offspring. High levels of mutation may create diverse but

unfeasible paths. Table 3.3 shows that the number of mutation nodes for two new child paths is one node per path. Thus, mutation is performed on a single node per path only. This causes higher probability of mutation for shorter paths. This means that the mutation process has a larger impact on the new offspring. On the other hand, longer paths have a lower probability of mutation.

The three chosen methods of performing single point path mutation are internal node, external coordinate and node omission mutation. Table 3.3 shows the probability that each mutation is applied. In this study, each mutation method is given an equal opportunity to run because they each have their own benefits. Thus, all three mutation methods are run in series and repeated again in a loop. Each mutation operator is applied at a rate of 1/3 for each generation. The first method that is applied is the internal node mutation as shown in Figure 3.19. Here, two random nodes, t_{m1}, t_{m2} are extracted from both offspring and exchanged in placements. In this case, the replacement nodes are obtained internally from both offspring paths.

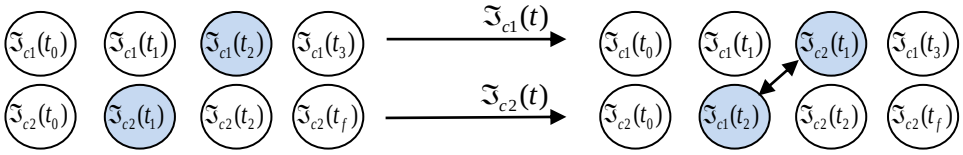


Fig. 3.19. Genetic Algorithm internal node mutation with selected parents.

$$\Im_{c1}(t) = [\Im_{c1}(t_0,...,t_{m1-1}) \ , \ \Im_{c2}(t_{m2}) \ , \ \Im_{c1}(t_{m1+1},...,t_f)] \tag{3.6}$$

$$\Im_{c2}(t) = [\Im_{c2}(t_0,...,t_{m2-1}) \ , \ \Im_{c1}(t_{m1}) \ , \ \Im_{c2}(t_{m2+1},...,t_f)] \tag{3.7}$$

s.t. $t_m \neq t_0$, $t_m \neq t_f$

With the second method, a random node from offspring is selected and removed from the offspring as shown in Figure 3.20. An externally sourced node from the three-dimensional environment is set in its place. These external nodes, t_{e1}, t_{e2} must be within the user defined search radius range, $r_{external}$ and obstacle free space to be considered as an option. Table 3.3 shows that in this study, a search radius of only 50 meters is used for finding the external mutation node. This means that GA doesn't explore the full environment unlike MA-RRF. This Is because GA is meant to be a more refined planning algorithm that improves the initial MA-RRF paths. The initial MA-RRF path planning process has already fully explored the search area.

Both methods of internal and external mutation are complementary to the algorithm. Internal mutation maintains optimal nodes from both parents. External mutation is implemented here due to its ability to produce offspring that can differ from their parent paths. This is because the new mutation node will continue to promote free space exploration and diversity as the generations' progress.

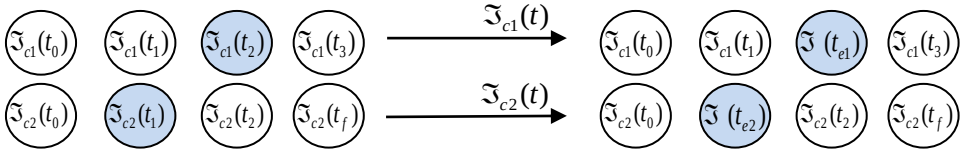


Fig. 3.20. Genetic Algorithm external node mutation with selected parents.

$$\mathfrak{I}_{c1}(t) = [\mathfrak{I}_{c1}(t_0,...,t_{m1-1}) \ , \ \mathfrak{I}(t_{e1}) \ , \ \mathfrak{I}_{c1}(t_{m1+1},...,t_f)] \tag{3.8}$$

$$\mathfrak{I}_{c2}(t) = [\mathfrak{I}_{c2}(t_0,...,t_{m2-1}) \ , \ \mathfrak{I}(t_{e2}) \ , \ \mathfrak{I}_{c2}(t_{m2+1},...,t_f)] \tag{3.9}$$

s.t. $x(t_{m1}) - r_{external} < x(t_{e1}) < x(t_{m1}) + r_{external}$, $x(t_{m2}) - r_{external} < x(t_{e2}) < x(t_{m2}) + r_{external}$

$y(t_{m1}) - r_{external} < y(t_{e1}) < y(t_{m1}) + r_{external}$, $y(t_{m2}) - r_{external} < y(t_{e2}) < y(t_{m2}) + r_{external}$

$z(t_{m1}) - r_{external} < z(t_{e1}) < z(t_{m1}) + r_{external}$, $z(t_{m2}) - r_{external} < z(t_{e2}) < z(t_{m2}) + r_{external}$

Lastly, the omission mutation is based on the deletion of a random node within both paths is shown in Figure 3.21.

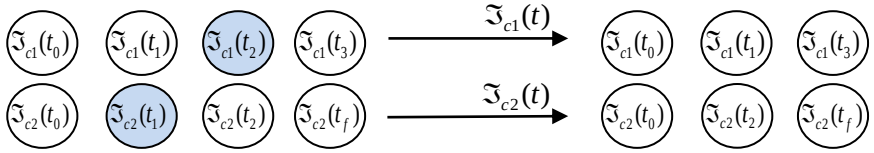


Fig. 3.21. Genetic Algorithm omission node mutation with selected parents.

$$\mathfrak{I}_{c1}(t) = [\mathfrak{I}_{c1}(t_0,...,t_{m1-1}) \ , \ \mathfrak{I}_{c1}(t_{m1+1},...,t_f)] \tag{3.10}$$

$$\mathfrak{I}_{c2}(t) = [\mathfrak{I}_{c2}(t_0,...,t_{m2-1}) \ , \ \mathfrak{I}_{c2}(t_{m2+1},...,t_f)] \tag{3.11}$$

s.t. $t_m \neq t_0$, $t_m \neq t_f$

The removal of a node from the offspring path encourages connections between nodes that were not initially immediately connected. Offspring paths that have lesser nodes will have larger distances between nodes thus the removal of a node creates significant impact. On the other hand, paths that have a larger number of nodes with smaller distances between them will experience fewer changes in direction. The omission mutation can also create unfeasible subsections within offspring. These subsections will require repair and encourage diversity between parents and offspring.

As shown in Figure 3.22, the offspring that are created after undergoing crossover and mutation are not collision free. These newly generated paths can contain path loops as well. Some node-to-node subsections within the child path may contain extreme bends that require the quadrotors to perform aggressive manoeuvrings. Thus, it is crucial that the resultant offspring paths undergo post processing. The algorithm attempts to repair the path subsections that contain collision points. This research reapplies the previously defined sampling based MA-RRF planner for path repair. It performs path repair by

bridging nodes that have any obstacles between them. This repair process allows the path population to maintain diverse offspring within its population.

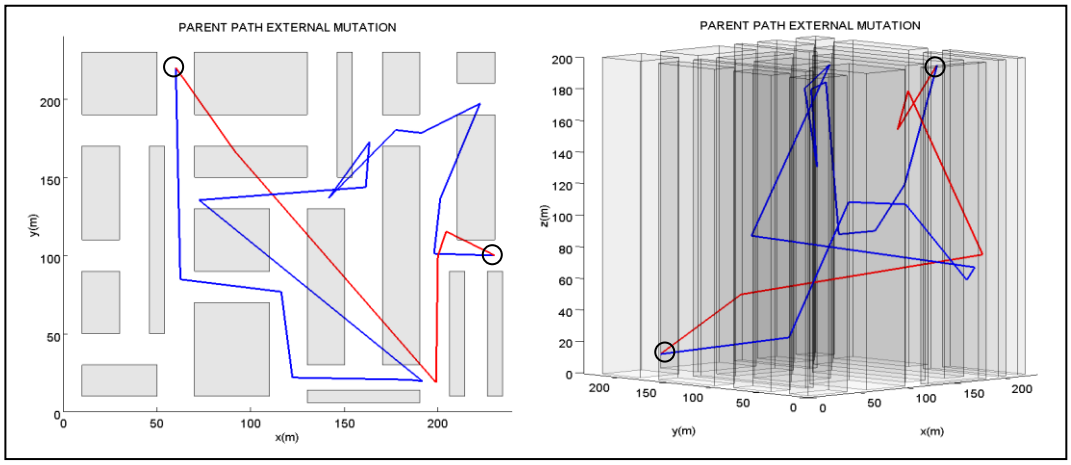


Fig. 3.22. Genetic Algorithm Child Paths External Mutation within Cityscape Environment.

3.3.6. PATH REPAIR & PRUNING

The child paths that were created by the GA's crossover and mutation operators can be filled with sections that collide with obstacles. There are a few methods that can be used to make these paths collision free. Through initial experimentation, the deletions of these unfeasible child paths prove to be extremely time consuming. This is due to the fact that collision free children are tough to produce organically. This is especially true within an environment with a high number of obstacles. This causes the GA to run for long periods in a bid to find feasible offspring paths. The second tried method is the removal of just the collision points within a child path. This is performed post the crossover and mutation stage. In most cases, it is the newly created collision filled waypoints that define the uniqueness of a child path. It is these waypoints that allow the offspring to be different from their parents' population. Thus, deleting these waypoints from the child path reduces its diversity. The diversity filter will test the waypoint differences between parent and child paths. The chances of the child path progressing to the next generation are greatly reduced when a diversity check is performed.

The last method requires a path repair algorithm. Many studies have performed path repair on unfeasible subsections within their designed paths. The MA-RRF planner is able to generate a feasible solution within seconds. This makes it the optimum algorithm for redesigning any path subsections that contain collision points. The MA-RRF algorithm is applied to reroute the path and successfully transform the child path to a collision free trajectory. Further experimentation shows that MA-RRF planner is the quickest and best method for producing and maintaining diverse paths within the

population. The MA-RRF path repair algorithm is run after the crossover and mutation process. Firstly, the root of the first tree is placed at the initial collision point. Next, the goal node is situated at the end of the collision point within the child path. Lastly, additional tree root points are spread across the corners of the environment. The MA-RRF algorithm is run for a set number of iterations. In this study, the repair algorithm is run for no more than 100 iterations. This allows the MA-RRF forest to quickly spread across the environment and search for alternate routes.

After the mapping process is complete, the algorithm tests if there are any forest branches that are close to the goal node. The path repair process is terminated once this branch is identified. In most cases, an alternative route is easily available. A new feasible path that bridges the collision points is extracted within seconds. The algorithm removes any nodes that are redundant. The offspring is then tested for its level of diversity and added into the population. In some cases, the path repair process terminates at 100 iterations. This happens before the planner is able to extract an alternative route. This means that bridging the collision points is too complex and requires a longer run time. In this case, the offspring is marked as unfeasible and removed from the population.

Figure 3.23 shows that only one offspring survives the path repair and pruning process. This method can be challenging to implement within highly complex environments with many obstacles. It can mean that no child paths are extracted if the repair process takes more than 100 iterations. In these cases, the termination point can be increased to a higher value to allow the MA-RRF trees to explore the environment further. In this study, the MA-RRF repair process has proven to be effective because it can repair enough paths to fill its population at each generation. This allows the algorithm for all variations of test spaces to continue to run for many generations. It also provides the same advantages over other algorithms as it did for the initial path planning process. This sampling based planner searches the entire test space randomly which speeds up the exploration process. Thus, quickly extracting feasible path nodes.

Next, this study applies a fully decentralized control system for multi-agent quadrotors. The trajectories that are designed by GA will be used as the input for the parallel run multi-agent control system. This system is a combination of the control system and mathematical model of each agent. This system will allow the end user to predict the motion and the derivatives of each quadrotor.

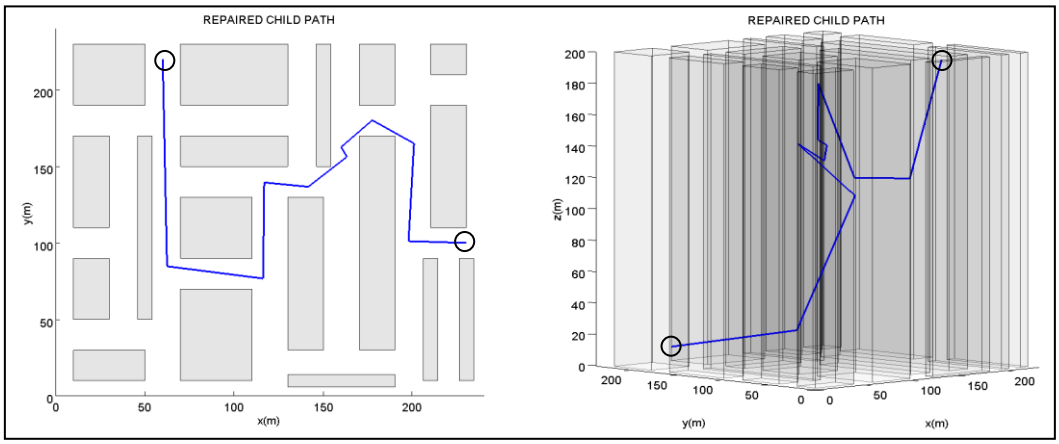


Fig. 3.23. Genetic Algorithm Repaired Child Path within Cityscape Environment.

3.4. MULTI-AGENT QUADROTOR CLOSED-LOOP CONTROL SYSTEM

In this research, a noncomplex nested control system is applied. This control system executes the simulated model for all agents in parallel on a multi-thread processing system. Thus, it reduces the collective simulation time for all agents simultaneously. Firstly, the paths that were generated for each agent by the GA path planner are converted into time based trajectories. This is achieved by implementing minimal jerk smooth splines. Then, the nodes within these smooth trajectories will be applied as the desired coordinates within each agent's control system. In this section, the structure of the individual quadrotor's control system is illustrated and simulated. The movement of each quadrotor will be estimated by its mathematical model and control system. This multi-layered control system is shown to adequately define the kinematics and dynamics of the quadrotor UAV.

3.4.1 MULTI-AGENT SMOOTH TRAJECTORY GENERATION

The paths that are generated by the GA process are not continuous and can contain extreme bends. These paths are transformed into minimal jerk trajectories by converting the paths into smooth splines as shown in Figure 3.24. Smooth splines are ideal for trajectory planning because it minimizes the effects of sudden changes across the trajectory. Path subsections can also be redesigned without having to replan the entire path. The end user has flexibility when creating splines of different orders. In many works, smooth splines are used to minimize the third derivative of the quadrotor's displacement. This produces minimal jerk trajectories. Similarly, the designer can generate UAV trajectories with minimal snap, crackle or pop by changing the highest degree of the spline equation.

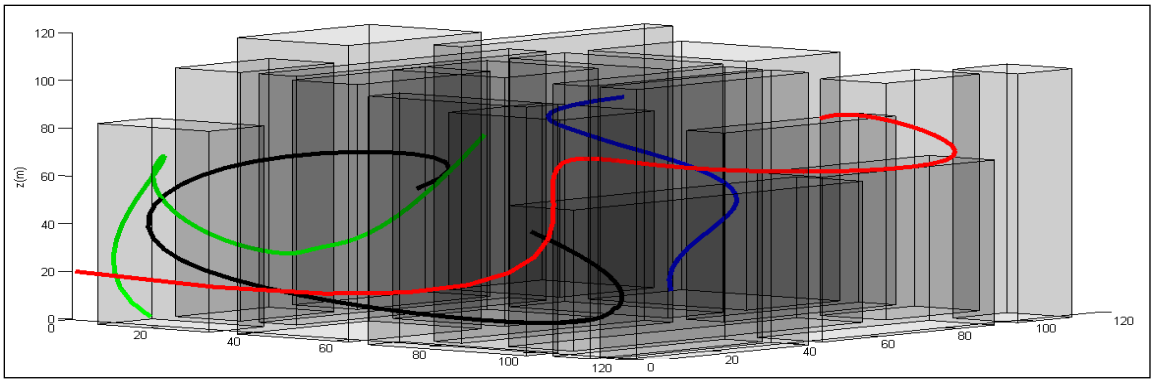


Fig. 3.24. Smooth splines for four agents across high rise cityscape environment.

This research focuses on long distance planning across large test environments. Each path contains multiple waypoints. Designing continuous trajectories that have many waypoints can be challenging. As shown in Figure 3.25, a breakpoint is where two curve segments meet. The continuity of a curve at a breakpoint describes if the quadrotors transition between a breakpoint with the same velocity or acceleration. Curves that lack in continuity can cause agent-to-agent collisions as well as an increase in path deviation due to aggressive movements. The transition between two waypoints can cause the agent to perform sudden movements. Thus, it is important that the agents move between the waypoints with minimal jerk and continuity.

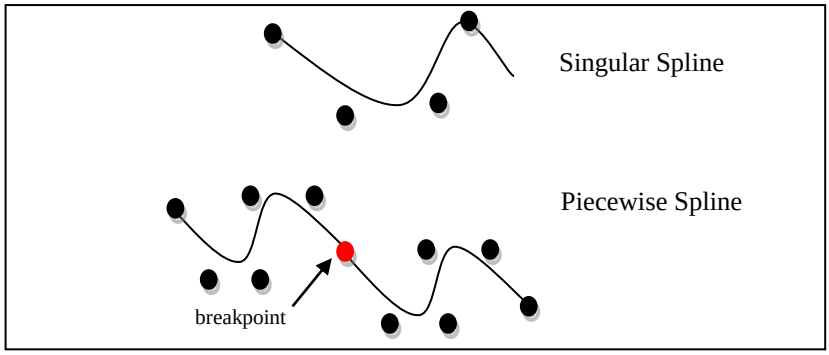


Fig. 3.25. Singular Spline, Piecewise Spline.

Here, fifth order splines are designed to minimize sudden jerks within each trajectory. Nonuniform splines are implemented so that the start and goal nodes of each path are preserved. These splines are constricted to adhere to continuity laws in order to generate smooth and accurate trajectories. The waypoints of each path are applied as the control points for the smooth splines. The number of control points is based on the number of nodes within the path. These control points are used as a guide when generating a smooth spline. Similarly, splines of fifth order will reduce overshoots within the quadrotor's control system. This will ensure continuity for the roll and pitch angles' second order derivatives. The smooth spine trajectory, $\chi(t)$ is obtained through,

$$\chi(t) = x_m B_{m,d}(t) + ... + x_0 B_{0,d}(t) = \sum_{q=0}^{m} x_q B_{q,d}(t) \tag{3.12}$$

s.t. $2 \le d \le m+1$

$t_{d-1} \le t \le t_{m+1}$

where $m+1$ corresponds to the number of control points

x_q are coordinates of the control points

$d-1$ the degree of the curve

t is the knot vector

$B_{q,d}$ is the basis functions of the curve

The basis functions are determined recursively. The value of the basis functions are based on the previously determined knot values. Initially, the basis function for the first degree is calculated by,

$$B_{m,1}(t) = \begin{cases} 1 & if \quad t_m \le t < t_{m+1} \\ 0 & otherwise \end{cases} \tag{3.13}$$

Next, the basis function for degrees that are larger than one is produced by the equation below,

$$B_{m,d}(t) = \left(\frac{t - t_m}{t_{m+d-1} - t_m} \right) B_{m,d-1}(t) + \left(\frac{t_{m+d} - t}{t_{m+d} - t_{m+1}} \right) B_{m+1,d-1}(t) \tag{3.14}$$

where $t_m = 0 \quad if \quad j < d$

$t_m = m - d + 1 \quad if \quad d \le j \le m$

$t_m = n - d + 2 \quad if \quad j > m$

These splines are highly adaptable based on the requirements of the end user. In this study, control points that are redundant to the spline formation are removed. Redundant point are nodes that do not contribute to a change in the direction of a quadrotor. This reduces the processing time of the multi-agent control system. One disadvantage of this process is it can create large distances between each waypoint. The closeness of a spline to its control points is often determined by the space between two waypoints. The further away the control points are, the further away the spline curve is from its boundaries. It is important to strike a balance between the waypoint-to-waypoint distances. Redundant nodes shouldn't be maintained but the waypoints must not be too far from each other especially around sharp corners.

Thus, some sections of the path are padded with more control points. In most paths, there are portions that require the quadrotors to perform aggressive turns. The aggressiveness of a node-to-node curvature is reduced by the addition of control points. This is performed by increasing the number of control points across sections that contain bends that are smaller than 90 degrees. The purpose of this adaptability is to reduce the possibility of safety zone breaches. This often occurs around the corners of obstacles as displayed in Fig.3.26(a). All the three of test environments in this study have many obstacles. The smooth splines must be able to avoid all obstacles. Firstly, the algorithm detects extreme curvatures within the paths that are designed by GA. Then, additional nodes are placed strategically around these bends as shown in Fig.3.26(b). This creates splines that follow the contour of the control points at turns.

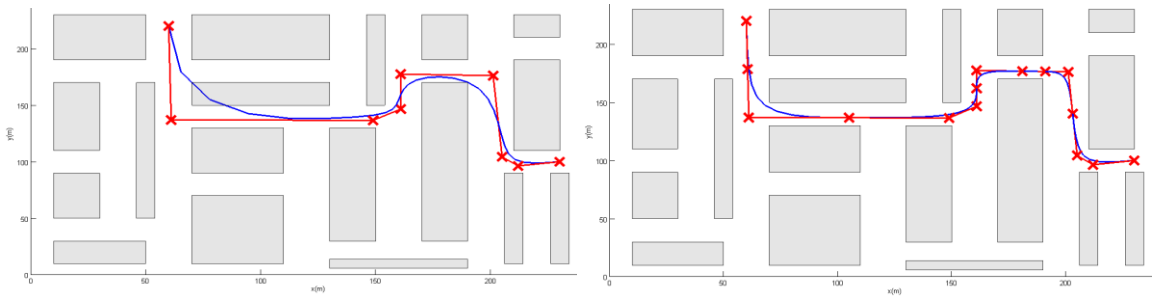


Fig. 3.26.(a) Non-adaptive smooth trajectory (b) Adaptive smooth trajectory.

The waypoints of the smooth trajectories are applied as the desired coordinates for the multi-agent control system. The closed-loop control system generates the estimated positional and rotational derivatives for each agent. This process shows the feasibility of the designed paths. The path planner must be able to produce paths that each quadrotor UAV is capable of tracking during flight. Paths that produce a large deviation error will face a higher chance of being removed from its population. This error will be determined in the next section through the difference in node-to-node distance between the predicted and planned path.

3.4.2 QUADROTOR MATHEMATICAL MODEL

The mathematical model that is shown below will be used to define the motion of a quadrotor UAV. This model will be used with a simple PD control system. This combination simulates a fast yet minimal error UAV controller. The nonlinear dynamic model in Figure 3.27 is derived under the following assumptions:

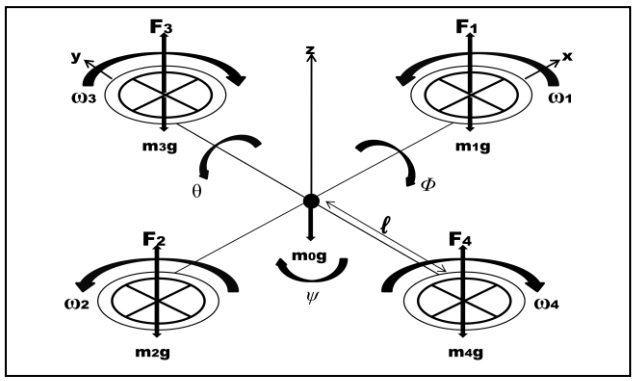


Fig. 3.27. Main structure of the Quadrotor.

a) The structure is rigid.
b) The structure is symmetrical about the centre of mass.
c) The centre of mass and the body fixed frame origin are assumed to coincide at the centre of the quadrotor's frame.
d) The propellers are rigid.
e) The thrust and drag are proportional to the square of the propeller speed.

The velocity vector, V consists of the quadrotor's linear velocity, v_B and its angular velocity, ω_B. The velocity values are obtained from the quadrotor's body frame,

$$V = [v_B \quad \omega_B] = [u \quad v \quad w \quad p \quad q \quad r] \tag{3.15}$$

Here, we employ Newton's second law of motion. It provides the equations for the net force, F_{net} and moment, M_{net} of the quadrotor's body frame:

$$F_{net} = \frac{d}{dt}[mv_B] + \omega_B \times [mv_B] \tag{3.16}$$

$$M_{net} = \frac{d}{dt}[I\omega_B] + \omega_B \times [I\omega_B] \tag{3.17}$$

where I represents the 3x3 identity matrix
m equals the mass of the UAV.

The inertial moments for the three-dimensional axes are defined as,

$$I = \begin{bmatrix} I_{XX} & 0 & 0 \\ 0 & I_{YY} & 0 \\ 0 & 0 & I_{ZZ} \end{bmatrix} \tag{3.18}$$

where I_{XX}, I_{YY}, I_{ZZ} = inertial moments.

The quadrotor creates angular motion about three axes, $[x\ y\ z]$ which are called the Euler angles, $[\phi\ \ \theta\ \ \psi]$. These angles are called the roll, pitch and yaw angles. The angular velocity of the quadrotor is determined by,

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.19}$$

Whereas, it's translational motion is produced across the three positional axes, $[x\ \ y\ \ z]$. The translational velocity of the UAV is represented by,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R^T \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.20}$$

The vector for linear accelerations that act on the vehicle's body frame, $\{B\}$ are transformed to inertial frame, $\{E\}$ through consecutive rotation matrices,

$$R = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi s\theta c\phi \\ s\psi c\theta & c\psi c\phi + s\psi s\theta s\phi & -c\psi s\phi + s\psi s\theta c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{3.21}$$

$$T = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \tag{3.22}$$

The system's equation (3.16) and (3.17) are then expanded to,

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} 0 \\ 0 \\ -F_z \end{bmatrix} + \begin{bmatrix} -gs\theta \\ gs\phi c\theta \\ gc\phi c\theta \end{bmatrix} + \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} \tag{3.23}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} \frac{M_x}{I_{XX}} \\ \frac{M_y}{I_{YY}} \\ \frac{M_z}{I_{ZZ}} \end{bmatrix} + \begin{bmatrix} \frac{I_{YY} - I_{ZZ}}{I_{XX}} qr \\ \frac{I_{ZZ} - I_{XX}}{I_{YY}} pr \\ \frac{I_{XX} - I_{YY}}{I_{ZZ}} pq \end{bmatrix} \tag{3.24}$$

The expansion of these equations can be viewed in detail within [140].

By expanding equation (3.20), the kinematic equations that define the translational flight of the structure are,

$$\begin{aligned}\ddot{x} &= (c\theta c\psi)\dot{u} + (s\theta c\psi s\phi + s\psi c\phi)\dot{v} + (s\theta c\psi c\phi + s\psi s\phi)\dot{w}, \\ \ddot{y} &= (c\theta s\psi)\dot{u} + (s\theta s\psi s\phi + c\psi c\phi)\dot{v} + (s\theta s\psi c\phi + c\psi s\phi)\dot{w}, \\ \ddot{z} &= (s\theta)\dot{u} + (-c\theta s\phi)\dot{v} + (-c\theta c\phi)\dot{w}\end{aligned} \tag{3.25}$$

The dynamic equations that portray the rotational movements of the quadrotor's structure are obtained from equation (3.19). As shown below,

$$\begin{aligned}\ddot{\phi} &= \dot{p} + (s\phi t\theta)\dot{q} + (c\phi t\theta)\dot{r}, \\ \ddot{\theta} &= (c\phi)\dot{q} + (-s\phi)\dot{r}, \\ \ddot{\psi} &= (\frac{s\phi}{c\theta})\dot{q} + (\frac{c\phi}{c\theta})\dot{r}\end{aligned} \tag{3.26}$$

The translational and rotational motion of the quadrotor is controlled by its four motors. The force and moments that act on the quadrotor are defined by equations (3.27) - (3.30). The quadrotor hovers when the speeds of all the rotors equal. Similarly, translational motion is created by increasing or decreasing the speed of the propellers. The total thrust that is generated by all four propellers is,

$$F_Z = k_b(T_1 + T_2 + T_3 + T_4) \tag{3.27}$$

Rotational motion is performed by changing the speed of just a pair of propellers. The rolling moment that is created around the x-axis is defined as,

$$M_x = k_b l(T_4 - T_2) \tag{3.28}$$

The pitching moment is defined as the angular movement around the y-axis,

$$M_y = k_b l(T_3 - T_1) \tag{3.29}$$

Lastly, the yawing moment is executed around the z-axis,

$$M_z = k_d(T_2 + T_4 - T_1 - T_3) \tag{3.30}$$

where l is the distance from the centre of gravity to the centre of the propeller
k_b is derived from aerodynamic contributions and
k_d considers air drag.
T_i Individual motor's thrust

Next, this open loop model is expanded to form the closed-loop control system. A robust PD controller is designed for the individual quadrotor. Here, the performance of the control system with both the full and simplified mathematical model is evaluated. The movements of the multi-agent quadrotors are analysed by obtaining these variables.

3.4.3 CLOSED-LOOP CONTROL SYSTEM DESIGN

Research on the quadrotor's control system have increased significantly in the last decade. The significance of this particular research as opposed to the others is its focus on multi-agent quadrotors. The basic singular quadrotor has a relatively simple structure and control system. Optimizing a multi-agent system can be much more complex. Whilst most multi-agent controllers perform satisfactorily, many are unable to cope with real life flights. This work also tests the planning and control of the quadrotor UAV within different environments and constraints. The designed control system must strike a balance between error minimization and speed of processing.

Here, a noncomplex nested control system is applied. This control system executes parallel simulation for all agents simultaneously on a multi-thread processing system. As displayed in Figure 3.28, the control system consists of three layers,

1) *Outer controller*: x translational controller and y translational controller
2) *Inner controller*: z translational controller, yaw rotational controller, roll rotational controller and pitch rotational controller
3) *Agent's system*: quadrotor's mathematical model

The smooth splines of fifth order, $\chi(s)$ are used as the input for the multi-agent control system. The output of the closed-loop estimator is the predicted flight paths for N_A agents. The predicted motion defines the dynamics of each agent's flight trajectory as segments between the initial, $\Im(t_{init})$ and goal, $\Im(t_{goal})$ nodes. The predicted flight path is defined as,

$$\Im(t) = \begin{cases} \Im(t_{init} \le t \le t_1) \notin \xi_{obs} \\ \vdots \\ \Im(t_{goal-1} \le t \le t_{goal}) \notin \xi_{obs} \end{cases} \tag{3.31}$$

where $\dot{\Im}(t_{init}) = 0$, $\dot{\Im}(t_{goal}) = 0$

The desired matrix is represented by the coordinates of the smooth spline waypoints,

$$r_d = \chi(t) = \left[x_d(t), y_d(t), z_d(t), \phi_d(t), \theta_d(t), \psi_d(t)\right] \tag{3.32}$$

Whereas, the current translational and rotational matrix is generated by the control system,

$$r = \Im(t) = \left[x(t), y(t), z(t), \phi(t), \theta(t), \psi(t)\right] \tag{3.33}$$

The control system is applied to minimize the positional error. The positional error is produced by the difference between the desired and current flight path. Thus, the error between desired and output states is defined by,

$$e_p = r - r_d \tag{3.34}$$

$$e_v = \dot{r} - \dot{r}_d \tag{3.35}$$

where e_p is the positional error

e_v is the error on velocity

The control system generates nondimensional speed control input, U for the quadrotor's mathematical model. The input signal for the motors will be used to estimate the movements of each quadrotor. The control signal is related to the brushless motor speeds through,

$$\left\lfloor F_Z \quad M_x \quad M_y \quad M_z \right\rfloor = [U_1 \quad U_2 \quad U_3 \quad U_4] \tag{3.36}$$

The first input computes the desired lift force for the quadrotor,

$$r_1 = k_{zp} e_{zp} + k_{zd} e_{zv} \tag{3.37}$$

$$U_1 = \frac{r_1 + mg}{c\phi c\theta} \tag{3.38}$$

The positional control along the x-axis and y-axis is controlled by the roll and pitch angular changes. The values are determined by,

$$U_x = k_{xp} e_{xp} + k_{xd} e_{xv} \tag{3.39}$$

$$U_y = k_{yp} e_{yp} + k_{yd} e_{yv} \tag{3.40}$$

The nested control signal from the x and y control system, U_x, U_y from (3.39), (3.40) produces the desired values for the roll and pitch controller, ϕ_d, θ_d.

$$\phi_d = -s\psi U_x + c\psi U_y \quad (3.41)$$

$$\theta_d = c\psi U_x + s\psi U_y \quad (3.42)$$

These are then applied as input for the roll and pitch control system,

$$U_2 = k_{\phi p} e_{\phi p} + k_{\phi d} e_{\phi v} \quad (3.43)$$

$$U_3 = k_{\theta p} e_{\theta p} + k_{\theta d} e_{\theta v} \quad (3.44)$$

Finally, the yaw control system is independently run,

$$U_4 = k_{\psi p} e_{\psi p} + k_{\psi d} e_{\psi v} \quad (3.45)$$

As previously stated, this mathematical model is applied to the control system. The closed-loop control system shows the translational and rotational changes of the quadrotor whilst it transitions along a planned trajectory. In this study, the control system is tested with a simplified version of the mathematical model in order to promote speed and efficiency. This model is defined in the next section. These characteristics are obtained without a large compromise on the accuracy of the quadrotor's mathematical model. The next section shows that the simplification of the model is necessary within a complex multi-agent UAV system.

3.4.4 INDIVIDUAL AGENT CONTROL SYSTEM

This section presents the nested control system. The chosen control system needs to create a balance between accuracy and simplicity. A highly complex multi-agent control system can affect the computing time immensely due to the increase in the number of agents. Both the MA-Spread and MA-Formation missions require parallel processing of four to eight agents. The control system for all agents must run simultaneously. In these cases, a noncomplex and minimal error system is best implemented. This study applies the constants in Table 3.4 during the MATLAB/SIMULINK simulation of the closed loop multi-agent quadrotors control system.

Based on Equations 3.23-3.24, the estimated translational and rotational movements of the quadrotor are related to the control signals in Equations 3.38 -3.45 by,

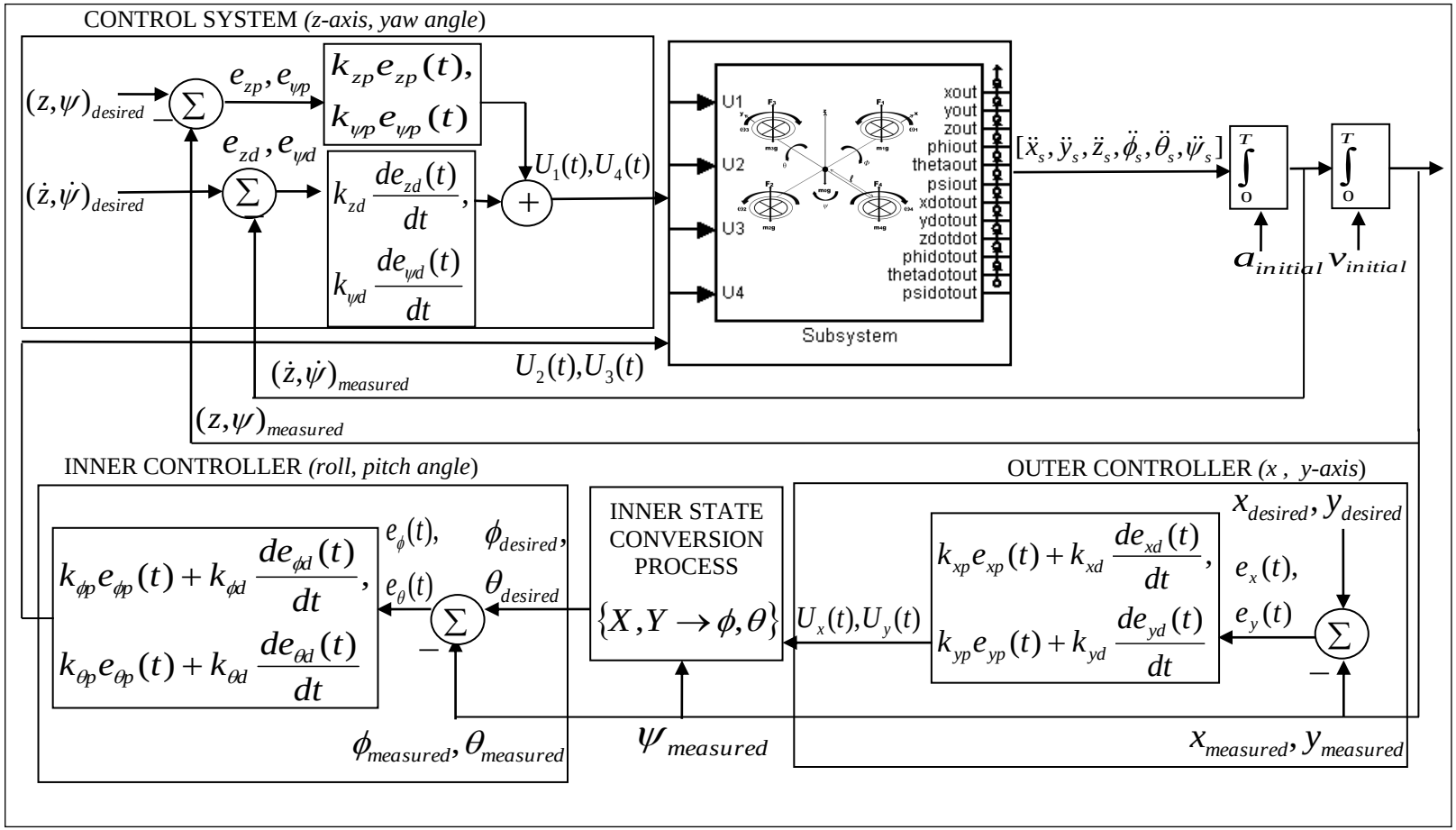


Fig. 3.28. Closed Loop Nested Control System.

TABLE 3.4. CONSTANT PARAMETERS

Parameter	Description	Value	Units
g	Gravity	9.81	ms^{-2}
b	Proportionality Constant	$3.13\text{x}10^{-5}$	
d	Drag	$9\text{x}10^{-7}$	
m	Mass	0.4794	kg
I_{XX}	x axis Inertial	0.0086	kgm^2
I_{YY}	y axis Inertial	0.0086	kgm^2
I_{ZZ}	z axis Inertial	0.0172	kgm^2
$[\,k_p\,,\,k_d\,]$	[proportional, derivative constant]	[0.14,0.08]	

$$\begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix}=\frac{1}{m}\begin{bmatrix}0\\ 0\\ -U_1\end{bmatrix}+\begin{bmatrix}-gs\theta\\ gs\phi c\theta\\ gc\phi c\theta\end{bmatrix}+\begin{bmatrix}rv-qw\\ pw-ru\\ qu-pv\end{bmatrix} \tag{3.46}$$

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix}=\frac{1}{m}\begin{bmatrix}\frac{U_2}{I_{XX}}\\ \frac{U_3}{I_{YY}}\\ \frac{U_4}{I_{ZZ}}\end{bmatrix}+\begin{bmatrix}\frac{I_{YY}-I_{ZZ}}{I_{XX}}qr\\ \frac{I_{ZZ}-I_{XX}}{I_{YY}}pr\\ \frac{I_{XX}-I_{YY}}{I_{ZZ}}pq\end{bmatrix} \tag{3.47}$$

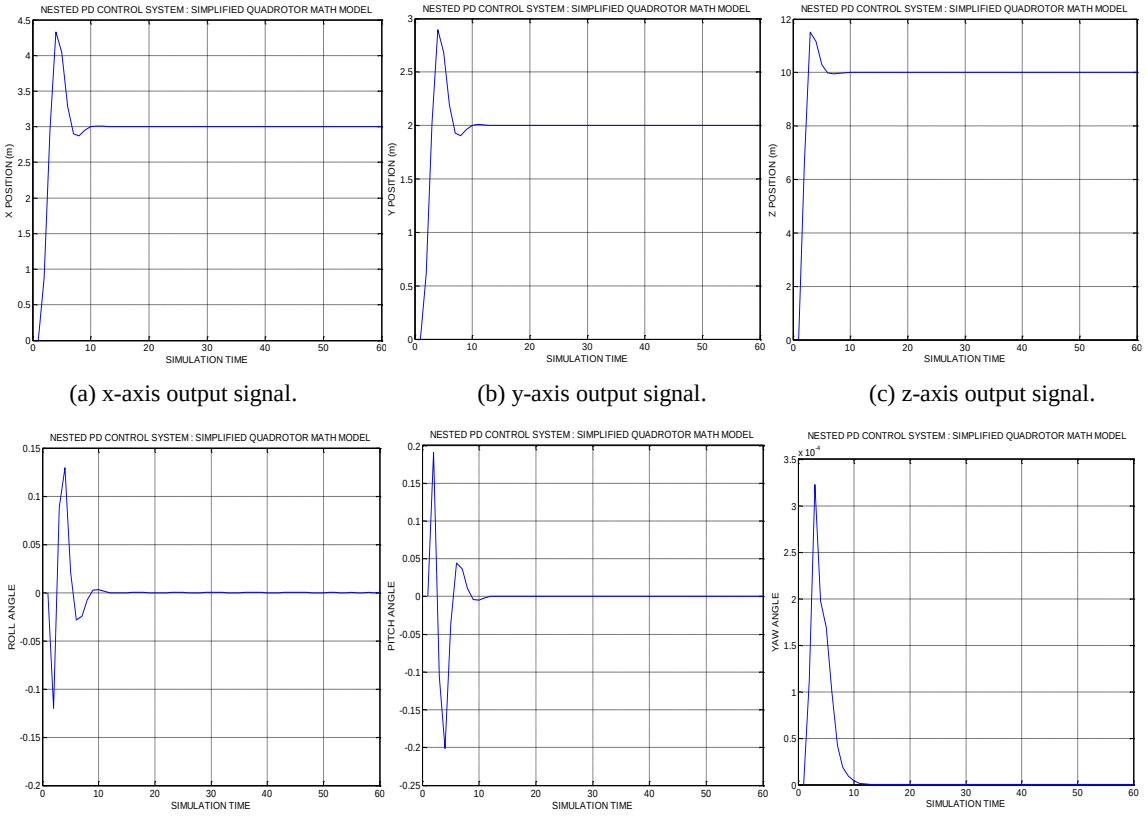
The mathematical model of a vehicle is simplified without sacrificing the accuracy of the quadrotor's movements. The simplification of the quadrotor model is often performed through the assumption of small rotational values where the Coriolis terms within equation (3.46) and (3.47) such as pr, pq and qr are assumed to be negligible. This assumption is implemented within this work because the paths are planned at a high resolution. The distance and angular difference between two waypoints are not large. Equations (3.25-3.26) that define the translational flight of the agents are simplified to,

$$\begin{aligned}\ddot{x}&=\frac{U_1}{m}(c\phi s\theta c\psi+s\psi s\phi)\ ,\\ \ddot{y}&=\frac{U_1}{m}(s\phi s\theta c\psi+s\psi c\phi)\ ,\\ \ddot{z}&=\frac{U_1}{m}(c\theta c\psi)-g\end{aligned} \tag{3.48}$$

Whereas, the dynamic equations that describe the rotational movements are similarly reduced to,

$$\begin{aligned}\ddot{\phi}&=\dot{\theta}\dot{\psi}\left(\frac{I_Y-I_Z}{I_X}\right)+\frac{l}{I_X}U_2,\\ \ddot{\theta}&=\dot{\phi}\dot{\psi}\left(\frac{I_Z-I_X}{I_Y}\right)+\frac{l}{I_Y}U_3,\\ \ddot{\psi}&=\dot{\phi}\dot{\theta}\left(\frac{I_X-I_Y}{I_Z}\right)+\frac{1}{I_Z}U_4\end{aligned} \tag{3.49}$$

The many-objective optimization algorithm often runs for a few generations before obtaining a well minimized set of solutions. At each iteration, its population represents multiple trajectories for many agents. Thus, the optimization algorithm makes many calls to the parallel control system. Figure 3.29 shows that the quadrotor UAV is able to reach its destination within 10 seconds.



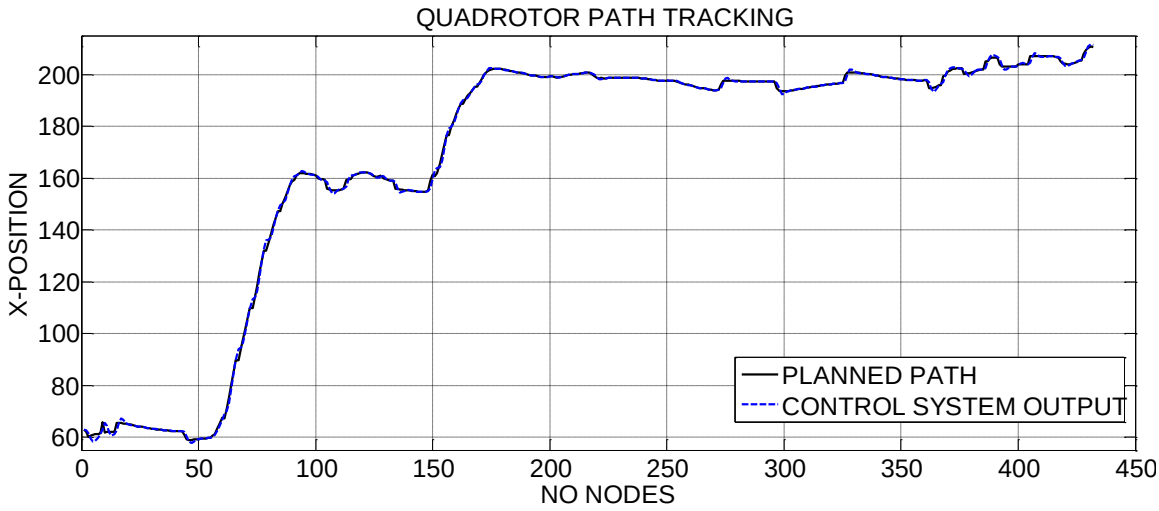
(a) x-axis output signal. (b) y-axis output signal. (c) z-axis output signal.

(d) roll-angle output signal. (e) pitch-angle output signal. (f) yaw-angle output signal.

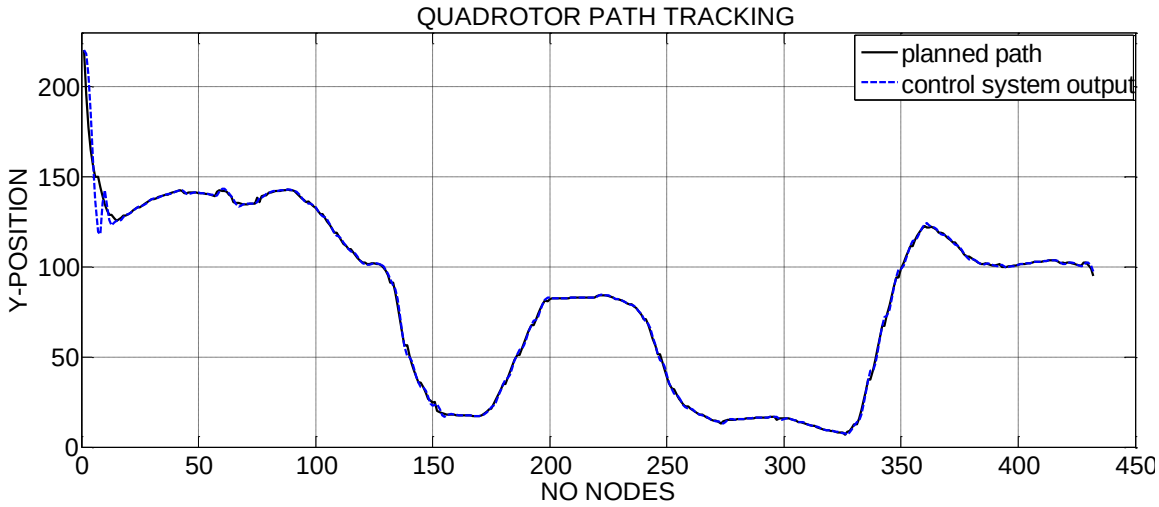
Fig. 3.29. Simplified model closed loop nested control system translational and rotational signal.

The speed of processing and minimal error of the simplified control system is ideal for this multi-agent quadrotor UAV system. This fast controller will also have an impact on the entire optimization algorithm because it will be utilized during many generations.

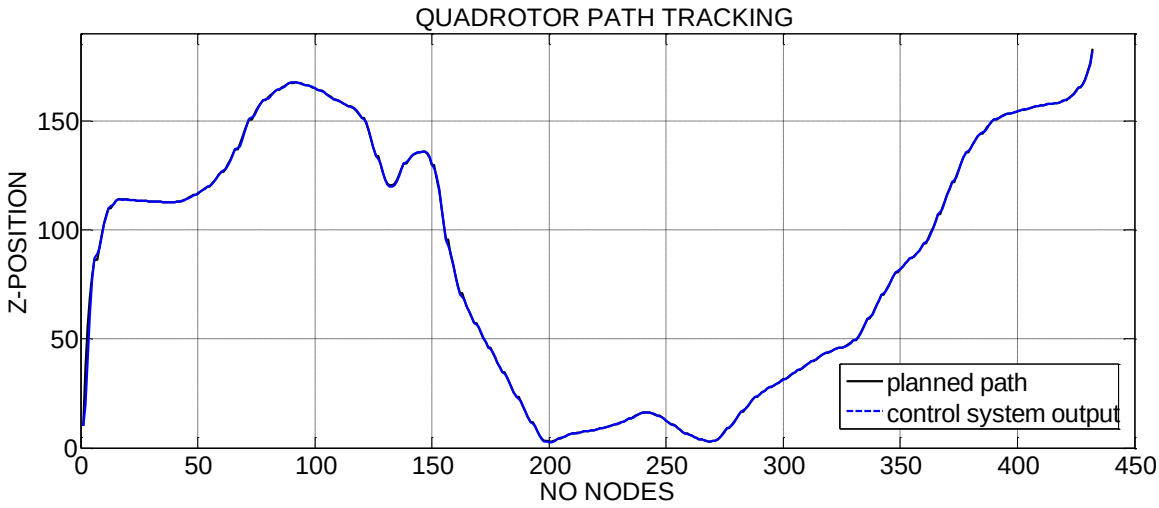
In this research, the simplified model of the quadrotor is used for the closed-loop control system. The quadrotor's rotational and translational output signal from its control system is shown in Figure 3.30-3.31. The path nodes across this path are spread no further than 2 meters apart. Here, Figure 3.30 shows the ability of the control system to track the planned path across a terrain. It shows that the output signals for all three positional axes can follow their desired coordinates with minimal error. Similarly, Figure 3.31 shows the roll, pitch and yaw signal output from the controller. The results show that the controller can keep up with the rotational turns across the path.



(a) x-position output signal.

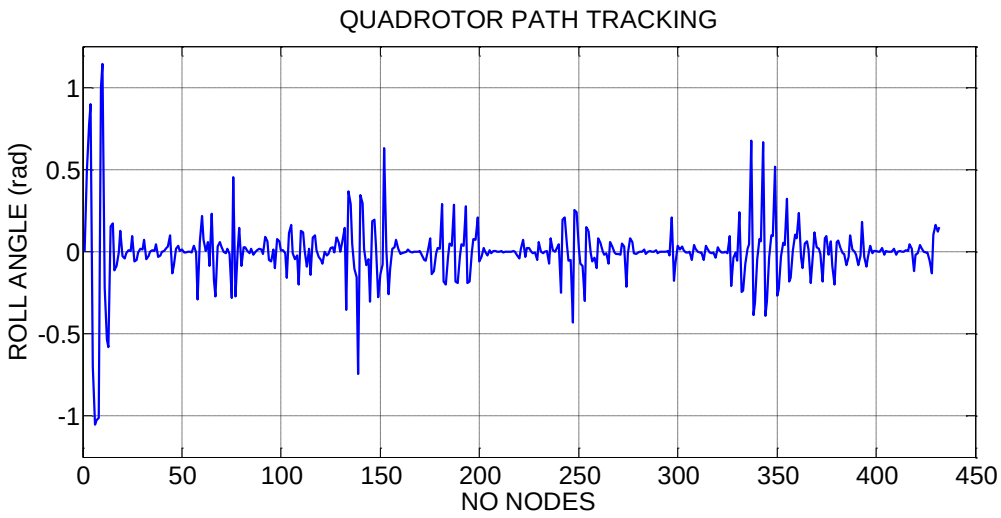


(b) y-position output signal.

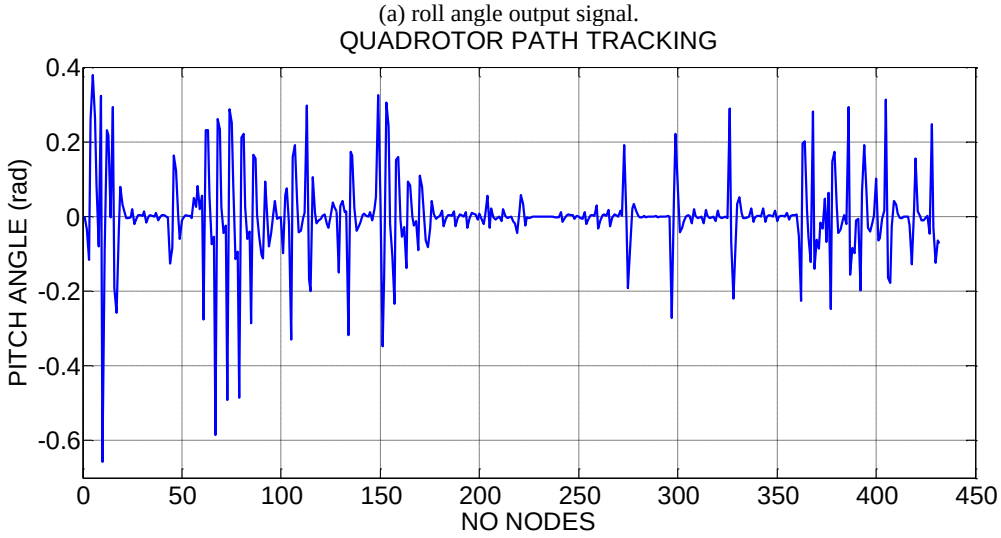


(c) z-position output signal.

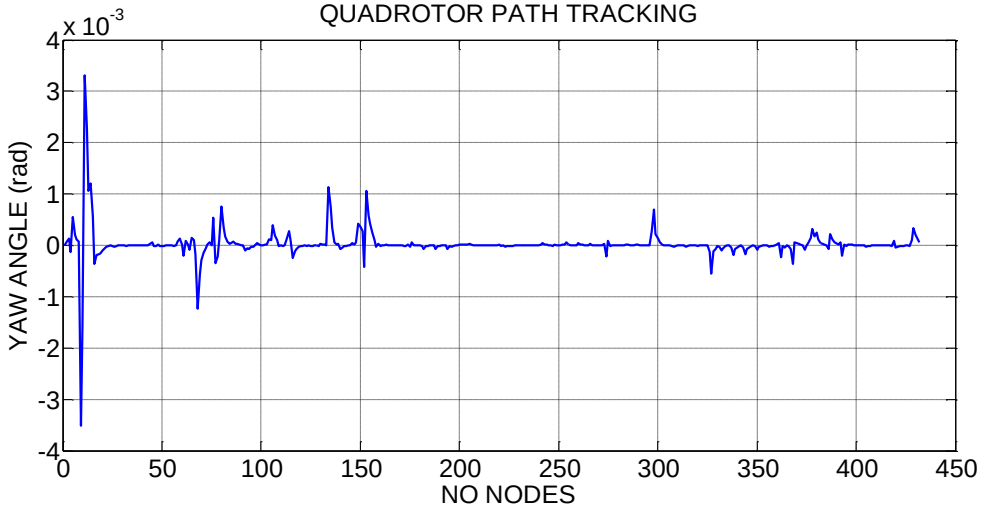
Fig. 3.30. Quadrotor's closed loop control system translational output signal.



(a) roll angle output signal.



(b) pitch angle output signal.



(c) yaw angle output signal.

Fig. 3.31. Quadrotor's closed loop control system rotational output signal.

3.5 SUMMARY

This research performs path planning for multi-agent quadrotor UAVs within three different test environments. Each environment holds a different set of challenges for the path planning algorithm. The first test space was the high-rise cityscape. The next space was the highly cluttered indoor environment. Lastly, a mountainous terrain was also used within this study. The results show that the obstacles within all of these test spaces were well mapped. Each obstacle had a safety boundary placed around them. The resulting images show that each environment required a different sized safety zones. This is because narrow roads can be closed off if the boundaries are too big. On the other hand, the agents can collide with the corners of the obstacles if the boundaries were too small. This shows that it is important to consider the characteristics of each environment before setting the value of any variable.

This chapter also showed the planning of the initial path population for four quadrotor UAVs. The MA-RRF algorithm was shown to be an expansion of the basic RRT path planner. The simulation results of each test space showed that the MA-RRF forest was able to quickly map the environments. It was capable of creating strategically placed forest links between the individual trees. The results also showed that the multi-agent sampling-based planner capable of quickly extracting tens of thousands of feasible paths. The MA-RRF multi-agent paths then underwent a diversity based filtration process. The findings within this chapter show that the MA-RRF planner has successfully created a diverse initial path population for the optimization process.

Next, GA was used to hybridize the MA-RRF paths and produce a diverse population of trajectories across all iterations. The results show that using the MA-RRF for repairing child trajectories can be highly beneficial. Four post processing GA operators were also implemented to further improve the survivability of these new child paths. Then, the path population was converted into smooth trajectories. These minimal jerk trajectories were used as the desired coordinates for the quadrotor UAV control system.

Lastly, this chapter focused on the modelling and testing of the quadrotor UAV's control system. The current chapter describes the important factors that must be considered when modelling the multi-agent quadrotors. Here, the mathematical model of the individual quadrotor was presented in its simplified form. The results showed that a fast and minimal error control system was chosen in comparison to a highly accurate one. The images also show that the output of the controller successfully predicts the control signals that define the collective speed and thrust of these quadrotors. It was able to estimate the translational and rotational movements of the quadrotor UAV. Collectively, MA-RRF, GA and closed-loop quadrotor UAV control system form a hybridized path

planner that continuously generates a tested collection of feasible trajectories for the optimization process within Chapter 4.

The next chapter shows the many objectives functions that are applied for both the MA-Spread and MA-Formation missions. The optimization of the multi-agent quadrotor UAV trajectories is performed as a team for both missions. The MA-Spread mission optimizes the combination of multi-agent trajectories. The objectives in the MA-Spread mission are dependent on the team's terrain exploration. On the other hand, the MA-Formation mission optimizes the formation reference trajectory at every generation. The estimated objective values are used by the Dimensionality Reduced Many-Objectives Optimization (DRMOO) algorithm to sort the trajectories based on their level of optimality and diversity. The trajectories that are ranked highest are maintained within the selection process and progress into the new generation of trajectories. The algorithm is designed to focus on the minimization of all objectives as opposed to mapping the Pareto front with full accuracy. This is the final step within the multi-agent quadrotor UAV trajectory planning and optimizing algorithm.

CHAPTER 4: TRAJECTORY OPTIMIZATION FOR MULTI-AGENT QUADROTOR UAVS

This research presents an offline long-range path planner that generates a large population of optimized trajectories for multiple autonomous quadrotors. It is capable of adapting to a variety of terrains, tasks and objectives. The missions that are typically undertaken by UAVs are generalized into two types. They are either defined as spatially distributed flight (MA-Spread) or dynamic formation flight (MA-Formation). This allows the end user to implement a standardized platform that is applicable towards many variations of real world multi-agent cooperative tasks. The first application is designed for missions that require independent multi-agent flight paths. This means that there is only a small amount of coupling between the neighbouring quadrotor UAVs. These agents have their own trajectories without being highly dependent on their neighbouring agent's flight direction. Formation flights are also a popular application for multi-agent quadrotors. In this case, the agents are highly dependent on their team mates. The trajectory for each agent is designed based on the formation's reference path and the desired formation shape.

The hybridized multi-agent quadrotor UAV trajectory planner and control system was presented in the previous chapter. They are both modified to suit the two variations of multi-agent systems. The hybridized MA-RRF and GA path planner is capable of generating a group of paths for the MA-Spread mission. It is also able to create a collection of formation reference paths for the MA-Formation task. This work shows that the same platform can be utilized even though both of these missions are the opposite in terms of agent-to-agent coupling. Thus, it can be used for missions where the agents are highly dependent on each other to complete a mission. It can also be applied for missions where the agents fly independently to accomplish a task. A common theme between these two differing missions is that they require solutions that consider the multi-agent system as a singular entity. The algorithm must be able to generate trajectories with various objective trade-offs for all agents without discrimination. Some multi-agent trajectories may have better values for certain objectives in comparison to the other. The collective objective values of each team of agents will be used to rank the trajectories within the optimization process. This allows the resources of all agents to contribute towards the completion of a task. It is a more efficient path planning algorithm in terms of managing the resources of a team of agents.

The smooth paths that were designed by the MA-RRF and GA planner will be used as the input for the two missions that are tested within this research project. Firstly, the flights trajectories for the agents within the MA-Spread mission are generated independently. Here, it is the combination of all the agents' independent trajectories that

is optimized. The trajectories of the four quadrotors within each path combination is applied to the objective functions as a collective. Thus, it is the combination of the independent flight paths that is used within the optimization process. The best combinations of agent paths are chosen as parents for the next iteration. Secondly, the MA-Formation application optimizes the formation's reference trajectory at each generation. The MA-RRF and GA path planners are used to generate a collection of formation reference paths. Then, formation shapes are designed based on the amount of free space that is around each reference nodes. Finally, the trajectories for each agent is generated based on the reference nodes and its formation shapes. Paths with higher complexities in formation shapes will be removed from the population. On the other hand, paths with more simplistic formation changes are maintained as parents for the next generation.

In this chapter, the individual agent's closed-loop control system will be expanded into a multi-agent control system for both missions. The data that is generated by the control system will be used to predict the values of the sixteen objectives. In these missions, different constraints need to be considered to simulate complex real-life multi-agent flights. In many cases, a selected number of objectives are considered simultaneously. These objective functions are often based on the dynamics and limitations of the quadrotor UAVs. It can also take into consideration the constraints of its environment and chosen mission. This chapter defines the mathematical equations for the many objective functions that are implemented in this research project. This study assumes that some of the sixteen objectives are conflicting in nature. Initially, we present the standard objective functions that are applied for both MA-Spread and MA-Formation scenarios. Next, the objective functions that are designed for multi-agent quadrotor independent flight, MA-Spread are defined. Lastly, the objectives that are utilized within the multi-agent quadrotor formation flight application, MA-Formation is shown.

Lastly, this chapter defines the parameters within the Dimensionality Reduced Many Objectives Optimization (DRMOO) algorithm. The DRMOO is used to rank the hybridized trajectories that are produced by GA and MA-RRF according to their level of optimality and diversity. The values of the many objective functions will be estimated with a closed-loop control system. Then, DRMOO will be used to rank the trajectories based on their estimated objective values. The aim of the DRMOO algorithm is to produce a final population of multi-agent trajectories that contain various strengths and weaknesses for these differing terrains. It is important that all of the many objectives are considered to be equally essential. In the next chapter, the results show that the DRMOO algorithm can be implemented as a standard platform for any multi-agent scenario or application.

4.1 MULTI-AGENT QUADROTOR OBJECTIVE FUNCTIONS

This section presents a standardized definition of the multi-agent quadrotors' physical constraints. It also takes into consideration the mission limitations such as environmental factors. These eight objectives are applicable to both the MA-Spread and MA-Formation missions. The constraints and limitations are represented by a collection of objective functions. These objective functions are not wholly based on the mean values between agents as with swarm or consensus theory. In this study, the definition of each objective function is dependent on the data that it represents. It is also not combined into a sum of aggregated data between all agents. Here, each objective is considered to be equally important. These objective functions are easily applicable to a variety of environments. It is changeable to accommodate many physical variations of the quadrotor. The end user can also consider other modes of communication, sensors, control methods and measurement units.

Firstly, the paths that are generated by the GA path planner will be converted into time based trajectories. Then, the nodes of these trajectories will be used as the input for the multi-agent control system. The control system outputs the estimated flight trajectories for all N_A agents simultaneously. The dynamics of the output trajectory by the control system is defined by,

$$\Im(t) = \left[x(t), y(t), z(t), \phi(t), \Theta(t), \psi(t)\right] \tag{4.1}$$

t = node number across smooth spline trajectory

The predicted flight path is split into node-to-node segments between the initial, $\Im(t_{init})$ and goal nodes, $\Im(t_{goal})$,

$$\Im(t) = \begin{cases} \Im(t_{init} \le t \le t_1) \notin \xi_{obs} \\ \vdots \\ \Im(t_{goal-1} \le t \le t_{goal}) \notin \xi_{obs} \end{cases} \tag{4.2}$$

where, $\dot{\Im}(t_{init}) = 0$, $\dot{\Im}(t_{goal}) = 0$

Each agent's positional states and derivatives will be used to predict the values of the many objective functions. As previously defined in Section 1.4, there are eight standard objective functions that are applied for both the MA-Spread and MA-Formation applications. These eight standard objectives for both missions are defined below:

When designing trajectories for a multi-agent UAV system, the most important criterion for end users is the flight distance from the defined initial to goal node. The first objective, ς_1 is designed to maintain the shortest paths.

$$\varsigma_1(t) = \sum_{1}^{A} \int_{t_{init}}^{t_{goal}} d_{\Im_A}(t) \quad dt, \forall A = \left\{ 1,...N_A \right\} \tag{4.3}$$

s.t $\quad \dot{\Im}_{\min} \le \dot{\Im}(t) \le \dot{\Im}_{\max}$

where $d_{\Im_A}$ = segment length

The second objective, ς_2 aims to maximize trajectories that hold many nodes at lower heights of the terrain. In this case, the reduction of flight time by all agents is prioritized. This cost function is adjustable based on the preference of the end user. The equation is easily modifiable to preserve higher nodes if flights at elevated altitudes are preferred. This option can be used by end users that require a larger view of the terrain.

$$\varsigma_2(t) = \frac{1}{N_P} \sum_{1}^{A} \int_{t_{init}}^{t_{goal}} (z_A(t) - h_T(t)\ dt, \ \ \forall A = \left\{ 1,...N_A \right\} \tag{4.4}$$

where h_T = terrain height

N_P = number of nodes within flight trajectory

Most studies implement cost functions that evaluate the trajectories of all agents as a whole. These objective functions are often used to assess the entire path from the initial to destination node. On the other hand, objective ς_3 optimizes the node-to-node progression. This allows the user to view the advantages and disadvantages of a path based on smaller subsections. This is performed by testing if two continuous nodes deviate further from the goal node. Paths that advance directly towards its goal as opposed to taking longer routes are maintained across generations.

$$\varsigma_3(t) = \sum_{1}^{A} b_{\varsigma_3}(t)\ dt \tag{4.5}$$

s.t $\quad b_{\varsigma 3} = \begin{cases} 0 \ , if\ d_{\varsigma 3}(t) < 0 \\ 1 \ , otherwise \end{cases}$

$$\text{where } d_{\varsigma 3}(t) = \int_{t=t_{init}}^{t_{goal}} \sqrt{(\Im_A(t_{goal}) - \Im_A(t))^2} - \sqrt{(\Im_A(t_{goal}) - \Im_A(t-1))^2}\ dt \tag{4.6}$$

Quadrotor UAVs have the advantage of manoeuvring across path curvatures with agility and speed. An adaptive cost function, ς_4 allows the end user to rank less aggressive

trajectories over paths that require extreme manoeuvring. The user can easily modify this objective function to ensure minimum jerk, snap or pop by changing the degree of the derivative. This study chooses to minimize the state derivatives of a trajectory as opposed to the path curvature to obtain smoother paths. It can be changed to maximize fast turns if flights at maximum velocity are preferred.

$$\varsigma_4(t) = \frac{1}{N_P} \sum_{1}^{A} \int_{t=t_{init}}^{t_{goal}} \left(\frac{d^{\,n} \Im_A(t)}{dt^n} \right)^2 dt, \;\; n = 3:6, \;\; \forall A = \left\{ 1,...N_A \right\}$$

(4.7)

where n =degree of smoothness

One way of measuring the feasibility and real-life applicability of the designed path is through its estimated trajectory error. The cost function, ς_5 compares both the GA planned trajectories, $\chi_A(t)$ with the predicted flight path, $\Im_A(t)$ of the quadrotor, *A*. Estimating the motion of the heterogeneous multi-agent quadrotors is accomplished through the application of a closed-loop control system. One input node from the GA planned trajectory equals to one output node from the control system. Thus, both variables are of the same type and same parameterization. The system consists of a nested PD control system and the mathematical model the different sized quadrotors. This function tells the user that if the designed paths have a high spline deviation, it requires the addition of more midpoint nodes across the path. This shortens the node-to-node distances and minimizes the deviation error.

$$\varsigma_5\ (t) = \frac{1}{N_P} \sum_{1}^{A} \int_{t_{init}}^{t_{goal}} \sqrt{(\Im_A(t) - \chi_A(t))^2} \quad dt, \;\; \forall A = \left\{ 1,...N_A \right\} \tag{4.8}$$

Obtaining time optimal trajectories can be a priority for end users with time constraints. The objective, ς_6 defines the optimality of a designed path by comparing two values. The first value is the estimated node-to-node controlled flight velocity, $v_{A,est}$ of the designed smooth spline. The second value is the direct waypoint-to-waypoint flight, d_{l_A} time at estimated optimal velocity, $v_{A,optimal}$. The optimal velocity is different from the maximum velocity of a quadrotor. The estimated optimal velocity for an agent across a path is obtained from the output of the agent's control system. This value changes every time the control system applied for each trajectory at each generation. This is because some paths may have complex turns and movements which causes the agents to move at different velocities. The ideal velocity changes based on the complexity of a path. In this study, the optimal velocity is estimated by obtaining the highest velocity value across the

entire path. The quadrotor's flight time from a direct node-to-node flight at ideal speed is compared to its estimated flight time across a smooth spline to test if the agent is able to fly to the best of its ability. Here, paths that promote flights close to the ideal flight velocity are preserved within the population.

$$\varsigma_6(t) = \sum_{1}^{A} \left[\int_{t_{init}}^{t_{goal}} (d_{l_A}(t)/v_{A,optimal})\ dt \Big/ (d_{\Im_A}(t)/v_{A,est}(t))\ dt \right], \forall A = \{\ 1,...N_A\ \} \quad (4.9)$$

where $d_{\Im_A}$ = smooth spline node to node trajectory distance

$v_{A,optimal} = v_{A,x-axis}(\max)$

The next objective, ς_7 functions to determine the flight time of a multi-agent trajectory combination. Quadrotors can fly fast and lift loads. These loads come in the form of measurement units, imaging, networking and sensory electronics. Efficient use of fuel is necessary to promote longer flight time.

$$\varsigma_7(t) = \sum_{1}^{A} \int_{t_{init}}^{t_{goal}} f_A(t) - f_A(t-1), \forall A = \{1,...N_A\} \quad (4.10)$$

f(t)= estimated node-to-node flight time from agent's control system

The avoidance of collisions with obstacles is another important criterion for autonomous flights across challenging terrains. Cost function ς_8 detects the number of no-fly zones, $\varepsilon_{k,boundary}$ that are breached. This function is used to minimize the risks of agent loss due to avoidable obstacle collisions.

$$(\xi_{k,} - \Delta d_{\xi}) \le \varepsilon_{k,boundary} \le (\xi_{k,} + \Delta d_{\xi})\ ,\ \forall k = 1,....N_{obs} \quad (4.11)$$

where Δd_{ξ} = obstacle boundaries buffer range

ξ_k = obstacle planes

The objective function defines the number of safety zones that have been breached by the entire team of quadrotors. Trajectories that require the agents to fly too close to the obstacles are slowly filtered out of the population.

$$\varsigma_8(t) = \sum_{1}^{A} \int_{t_{init}}^{t_{goal}} b_{\varsigma 8}(t)\ dt, \forall A = \{\ 1,...N_A\ \} \quad (4.12)$$

$$\text{s.t.} \quad b_{\varsigma 8} = \begin{cases} 0 \ , if \ \Im_A(t) \notin \varepsilon_{k,boundary} \\ 1 \ , otherwise \end{cases}$$

As previously defined, these eight objective functions and constraints are applicable for both independent, MA-Spread and dependent, MA-Formation UAV flight missions. The end user is also given the option of including costs that are specific to either the MA-Spread or MA-Formation mission. The objective function within each mission is dependent on the costs that are considered by researchers that study multi-agent UAV flights. Spatially Spread flights are often used for missions that require the agents to Spread across the environment and collect as much data as possible. On the other hand, formation flights are used for payload transportation or target tracking. This gives the end user the flexibility of using the objective functions that are suitable for their mission.

4.2 MA- SPREAD APPLICATION

Multi-agent unmanned aerial vehicles (UAV) systems are frequently used for complex missions such as search and rescue, reconnaissance, terrain mapping, wildlife research, target tracking as well as forming ad hoc wireless networks. Today, quadrotors can be purchased for a low cost and modified to transmit data from many types of sensory systems. It is easily expandable to a large sized multi-agent system. The popularity of the quadrotor within the consumer market is due to its ability to hover and deliver clear imagery in real time. It has the ability to undertake aggressive turns and capture videos at high definition. These agents are currently being used by media content creators, journalists, scientists, delivery companies, governmental bodies as well as hobbyists. Implementing a multi-agent system means that there is potential that the same mission can be completed at a much faster rate.

This research aims to analyze and implement a standardized platform for simultaneous multi-agent trajectory generation and optimization algorithm that is applicable towards many variations of real world multi-agent cooperative tasks. Missions typically undertaken by independent UAVs are generalized as spatially distributed flight scenarios. This hybridized algorithm is designed to produce feasible smooth trajectory solutions for a multi-agent system. It takes into consideration the many objectives that are based on the purpose and constraints of the MA-Spread mission. Here, we present the objectives that are specifically designed for Spread flight. Performance analysis of the hybridized algorithm is presented through the flight simulation of the multi-agent quadrotors. These independent agents are Spread across multiple wide-area test environments.

4.2.1 MA- SPREAD TRAJECTORY COMBINATIONS

In this research, the optimization of a group of agents is performed by applying the team as one entity. The agents in the spatially distributed cooperative flight missions are often supplied with their own trajectories. Here, the best path that is assigned to an individual agent may introduce shortcomings to its neighbouring agents. The best path for an agent can cause possible inter-agents collisions with other agents. This causes the other agents to take longer routes to avoid any collisions. There is also a chance that the entire team could return with minimal collective terrain exploration. This is because the agents may fly over similar areas. One can assume that the best path for an individual agent may not necessarily lead to minimum cost values for the entire team. It is important to generate the well minimized paths for all agents collectively. This ensures that the cost values for the team as a whole is optimized.

Therefore, optimization within the MA-Spread application is performed for a collective set of paths. The paths that are generated by GA are used for various combinations of the multi-agent's flight paths. A set number of path combinations are created at each generation. The number of combinations is based on the population size of the optimization algorithm. The level of diversity between the combinations is based on the number of unique trajectories that are maintained within an agent's path population. The possible number of path combinations is equal to the binomial coefficient,

$$\binom{N_{\Im}}{N_c} = \frac{N_{\Im}(N_{\Im}-1)...(N_{\Im}-N_c+1)}{N_c(N_c-1)...1} \tag{4.13}$$

where $N_{\Im}$ = number of paths per agent of current generation

N_c = number of combinations

This research tests the MA-Spread application with a team of four quadrotors. Figure 4.1 shows the way each path combination is stored within the database. It is these randomized combinations, $C_i(t)$ of flight paths that are optimized for each generation. This process is designed to promote the maintenance of the well minimized collective trajectory within the final generation.

$$C_i(t) = \{\Im_1(t) \quad \Im_2(t) \quad \dots \quad \Im_A(t)\} \; ; \; \forall A = \{1, \dots, N_A\} \tag{4.14}$$

where i = number of multi-agent path combinations

A = quadrotor agent

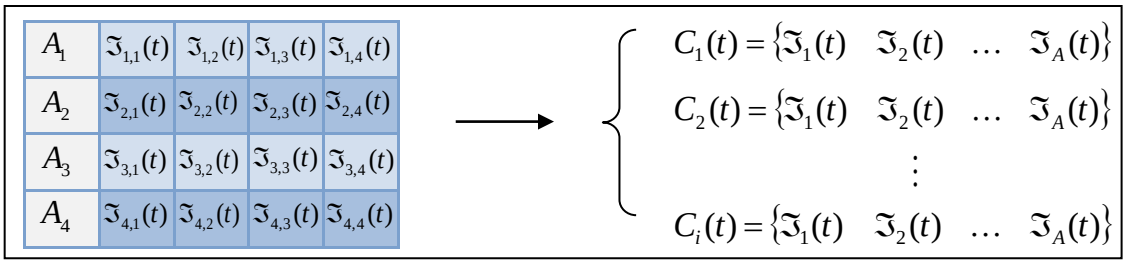


Fig. 4.1. MA-Spread randomized combinations of multi-agent flight trajectories.

Each path combination is applied one by one into the multi-agent control system. This study uses a population of 30 path combinations, *i*. The path combinations hold paths for four different quadrotors, *A*. The path nodes of each agent are defined as the desired coordinates for the closed-loop control system. As previously defined, the state derivatives for all four agents are obtained simultaneously through parallel processing on a multi-thread system. The estimated state values from the control system are then used as the input for the objective functions. The cost values for the current path combination are used by the many-objectives optimization algorithm for the ranking process. The combination of multi-agent's paths that are both diverse and well minimized is ranked higher than combinations that are suboptimal.

Lastly, the best combinations from previous generation are stored as the next generation's parent population. The database stores the paths from each combination according to its agent. Thus, the number of unique paths per agent is dependent on the combinations that survive the selection process. This means that the path database for an individual agent can contain duplicate paths. This study implements elitism within the GA's selection process. New path combinations are generated from the current parent population and the new child trajectories.

4.2.2 MA-SPREAD CONTROL SYSTEM

An expansion of the single agent's hierarchy is performed to include the parallel simulation of four quadrotor UAVs simultaneously. The multi-agent architectures show the impact of having a control system within an optimization algorithm. Both the input and output data for the control systems are extremely important because it determines the type of cost functions that can be implemented. The control systems are simulated with MATLAB/SIMULINK on a multi-thread processing system. The data from the control system flows from the lower level control loops to the higher-level trajectory planning algorithm. These different layers can be implemented for a group of heterogeneous quadrotor UAVs despite their differences in hardware or software. Thus, simplifying the framework of the multi-agent control system is beneficial for many real-world applications.

Figure 4.2 shows the information flow across the trajectory planning algorithm for the MA-Spread mission. In this study, four quadrotors, $i = 1:4$ will fly independently across a variety of test spaces. Initially, the control system requires the constraints and limitations of each agent. This can be obtained from the physical structure of the quadrotor's hardware and its electronics. As previously defined, free space mapping and path planning is performed by MA-RRF and GA. The free space mapping process produces data such as the obstacles and their boundaries, ξ_{obs}. It also divides the obstacle free three-dimensional space, $\xi \in \Re^3$ into grid blocks, G. The size of each grid block, V_g is predefined based on the size of the test space. An initial collection of paths, $N_{parent\,pop\,i}(t_0)$ is provided by MA-RRF as input for the GA. Independent paths are designed for all four agents without interference from the motion of their neighbouring agents. Each agent has a new collection of paths at each generation, $N_{newgen\,i}(t_{iter})$. These paths are converted into trajectories by creating minimal jerk splines, $\chi_i(t)$.

This information flow across the different subsections of the MA-Spread system will form the input for the multi-agent control system. Various path combinations, $C_j(t)$ are formed from the trajectories that were developed for each agent. These trajectory combinations are applied within the control system one by one. At each iteration, the run time of the control system is defined through the number of nodes within the current combination's longest trajectory. Next, the control system and mathematical model for each quadrotor is run in parallel. The data that is required by each agent's estimator are the desired coordinates and angular rotations. This information is required by the agent's control system. Each subsection of the control system is able to calculate and reduce the positional error of a quadrotor. The initial values of an agent's position and its derivatives are also required by the integrator within the estimator.

The secondary part of a control system is its output data. The information must be easily accessible and understandable to the end user. The estimator's output data is stored within a shared database after each trajectory combination has passed through the system. The values for each iteration are stored based on the path combination that was applied. This makes it easy for the end user to identify which path combination is the best choice at the end of the optimization process. The values that are directly obtained from the control system are the predicted flight path of each agent, $\Im_i(t)$ as well as its derivatives. Similarly, the predicted rotational values and its derivatives are also obtained straight from the estimator. The output of each subsection within the control system provides the estimated path deviation, e_p for each agent. Lastly, the predicted flight time, t_{flight} for the entire team is also attained at the end of the simulation process.

All of the output data is crucial for determining the values of the objective functions within the MA-SPREAD application. The estimated values will be used to predict the flight time of all the agents. It will also provide insight as to the number of possible agent-to-agent or obstacle collisions. The estimated flight paths show the aggressive turns that the agents are required to perform. The end user is most interested in the cost functions that are designed specifically for the MA-SPREAD mission. Thus, the control system tells the user about the multi-agent team's collective sensory coverage and overlap across each test space. It also provides the changes in the network topology between each agent during flight. These objective functions will allow the many objective optimization algorithm to rank the path combinations. The combinations are ranked based on their level of dominance, $N_{dominant}(t_{iter})$ and diversity, $nc\ (t_{iter})$. The ranking system, DRMOO will be elaborated in the end of this chapter.

4.2.3 MA-SPREAD OBJECTIVE FUNCTIONS

The data from the previously defined control system will be used to estimate the values of the 12 objectives that are within the MA-Spread mission. Four new objective functions are used in addition to the standard 8 objective functions. These four objective functions cater specifically to the application of Spread flight missions. Here, the optimization algorithm aims to maintain multi-agent paths that maintain connectivity whilst mapping the free space of the terrain. It also aspires to minimize agent-on-agent collisions and redundant sensory data. The parameters that are applied within the MA-Spread application are shown in Table 4.1 [141-142]. A team of four quadrotors is used within this mission. Two different quadrotor models are applied in this study to create a more flexible system. Both of these models are different in size and weight.

TABLE 4.1: MA-Spread PARAMETERS

Description	Value	Description	Value
Number of Flight Paths Combinations	30	Maximum Agent Velocity	$10ms^{-1}$
Number of agents	4	Number of Gaui 330X-S Agents	2
Redundancy Similarity Threshold	$0.75\,N_p$	Gaui 330X-S Agent's Size	$0.533m$
Sensory Overlap Terrain Block Size	$30m^3$	Gaui 330X-S Agent's Maximum Fuel	15mins
RF Network Range Threshold	$0.75\ \xi_A$	Number of Fyetech Agents	2
Safety Zone Obstacles Boundary	$\xi_{obs} \pm$ 6m	Fyetech Agent's Size	$0.705m$
Minimum Agent-to-Agent Distance	$10m$	Fyetech Agent's Maximum Fuel	10mins

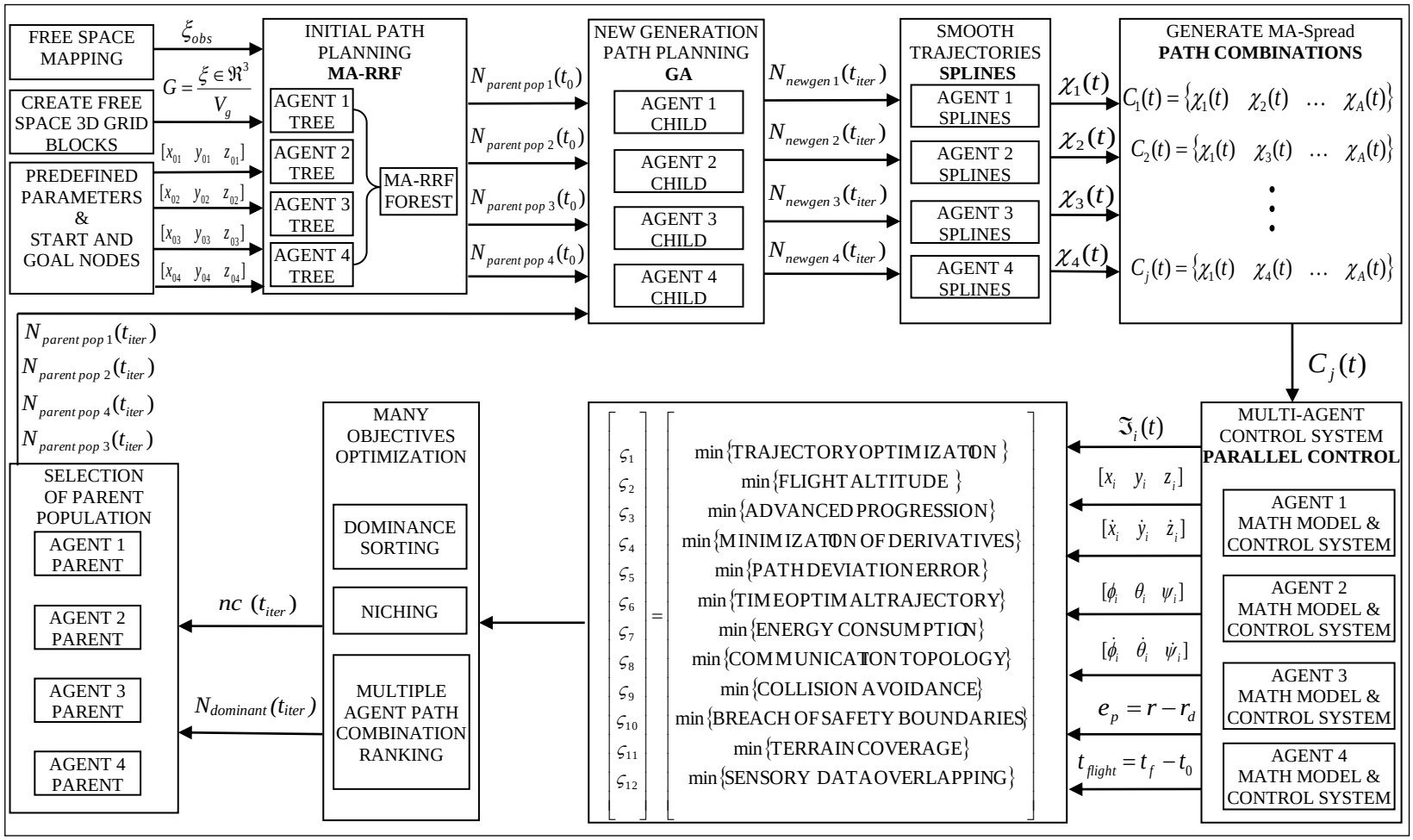


Fig. 4.2. MA-Spread closed-loop nested control system.

The first objective within the MA-Spread application is based on maintaining communication between the agents. Communication disruption occurs when the distance between two agents are further than the network signal range. The amount of signal decay grows larger as the agent-to-agent distance becomes longer. The effects of signal decay are such as data packet corruption or loss. Retransmission of these lost data can cause a delay in the prediction of possible collisions as well as shared free space mapping. Figure 4.3 shows the progression of four agents within the mountainous environment. In the first image, the distance between agents at each sample time is small. On the other hand, with the second image, there exists longer waypoint distances between the quadrotors that can cause network decay. This objective function attempts to minimize the network decay by reducing the distances between the agents.

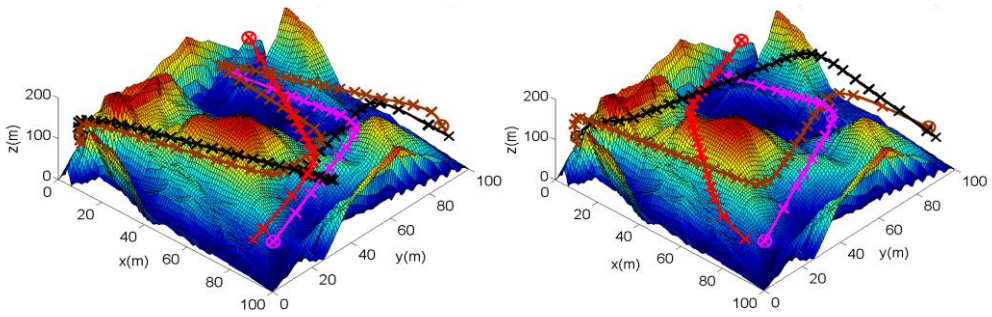


Fig. 4.3. Collective multi-agent paths with less and more network decay across mountainous terrain.

The network between quadrotor agents is defined through an adjacency matrix. The matrix entry a_{ij} is defined as,

$$a_{ij} = \begin{cases} \alpha, & if\ A_i \rightarrow A_j,\ \ \forall i, j = \{1,\ldots,N_A\} \\ 0, & otherwise \end{cases} \tag{4.15}$$

where $A_i \longrightarrow A_j$ = connection exists between agent *i* and agent *j*

s.t. $$\alpha = \begin{cases} 1, if\ d_{G,\min} \le d_{ij}(t) \le d_{G,\max} \\ 0, otherwise \end{cases}$$

Ideally, a fully connected graph, $G_f = \{V, A(t)\}$ is maintained when all agents are within the maximum network range, d_G. A well-connected system as shown in Figure 4.4(a) allows for minimal delay data exchange between shared database and the agents. Function, ς_9 serves as a measure of agent-to-agent connectivity. It is used to penalize trajectories that cause connection losses between the agents.

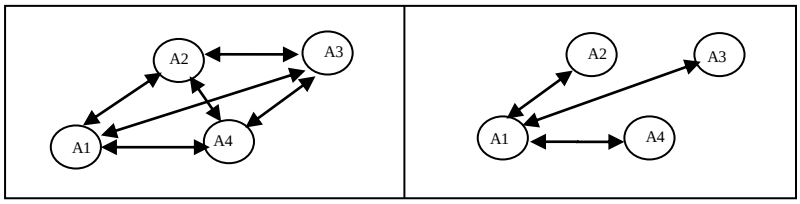


Fig. 4.4. Multi-agent (a) fully connected (b) partially network topology.

$$\varsigma_9 = \int_{t_{init}}^{t_{goal}} b_{\varsigma 9}(t)\ dt \tag{4.16}$$

s.t. $b_{\varsigma 9} = \begin{cases} 1\ , if\ a_{ij}(t) \neq G_f \\ 0\ , otherwise \end{cases}$

There can be possible collisions between agents during the simultaneous operation of a team. This can happen especially when the agents are all flying in different directions. Possible collision detection is estimated based on two criterions, the distance between agents, d_{ij} and the time of collision. The estimated position of each agent at a time period can be used to test if the agents are too close to each other.

$$d_{ij}(t) = \int_{t_o}^{t_1} x_j(t) - x_i(t)\ dt, \int_{t_o}^{t_1} y_j(t) - y_i(t)\ dt, \int_{t_o}^{t_1} z_j(t) - z_i(t)\ dt, \forall i = \left\{ 1,...N_A \right\}, \forall j = \left\{ 1,...N_A \right\} \tag{4.17}$$

s.t. $d_{ij,x\min} \le d_{ij,x}(t) \le d_{ij,x\max}\ ,\ d_{ij,y\min} \le d_{ij,y}(t) \le d_{ij,y\max}\ ,\ d_{ij,z\min} \le d_{ij,z}(t) \le d_{ij,z\max}$

This objective penalizes collective trajectories that hold possible collision points. It allows the end user to either replan the spline subsection or choose a collision free trajectory instead.

$$\varsigma_{10}(t) = \int_{t_{init}}^{t_{goal}} b_{\varsigma 10}(t)\ dt \tag{4.18}$$

s.t. $b_{\varsigma 10} = \begin{cases} 0\ , if\ d_{ij,\min} \le d_{ij}(t) \le d_{ij,\max},\ t_j - t_i \le t_{collision} \\ 1\ , otherwise \end{cases}$

Missions such as reconnaissance and surveillance require the agents to Spread across an environment. The agents within these missions collect a large amount of sensory data. Here, the maximization of uncertain terrain coverage is paramount. In this study, we assume free space mapping from a height with three-dimensional grid mapping. The number of grids, $N_{g,total}$ per environment is defined by,

$$G = \frac{\xi \in \Re^3}{V_g} \tag{4.19}$$

The amount of free space coverage by each agent defines the amount of sensory data that is collected by the entire team. Figure 4.5 shows the progression of four agents across the cityscape environment. In the first image, it is visually visible that the flight trajectories of all agents successfully explore a high percentage of the terrain. On the other hand, in the secondary figure, the agents fly across duplicate areas. This minimizes the amount of terrain coverage and reduces the amount of surveillance data that can be collected by the entire team. It can also cause an increase in the collection of redundant sensory data.

Cost function, ς_{11} prioritizes trajectories where the quadrotors fly across uncertain areas. This reduces flights across areas that have been well mapped across generations. Each grid block is given the value of zero initially. As the algorithm progresses, the value of each grid block, N_{grid} increases based on the number of times an agent flies across it. Thus, the minimization of this cost function maintains trajectories that have flown over environments that are more uncertain or hard to reach

$$\cdot\ \varsigma_{11} = \sum_{1}^{A} \int_{t_{init}}^{t_{goal}} \sum_{g=1}^{G} N_{grid}(g) + b_{\varsigma 11}, \forall A = \{1, \ldots N_A\} \tag{4.20}$$

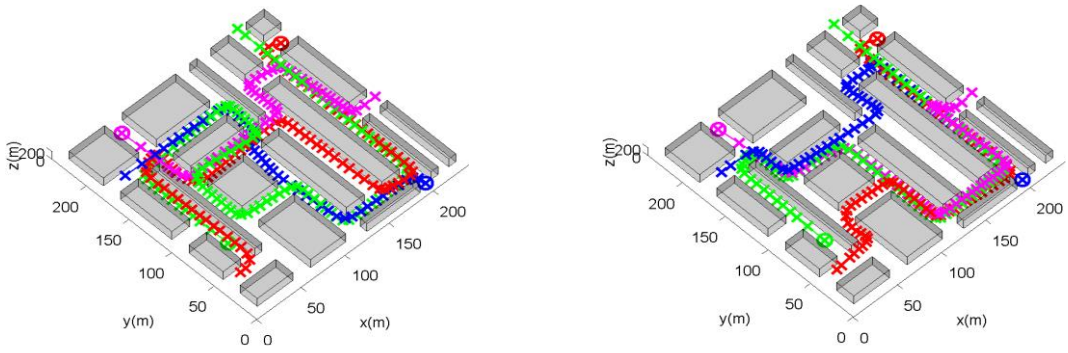


Fig. 4.5. Multi-agent high and low terrain coverage across cityscape environment.

$$b_{\varsigma 11} = \begin{cases} 1 & if \quad (x_{gl}, y_{gl}, z_{gl}) < (x, y, z) < (x_{gu}, y_{gu}, z_{gu}) \\ 0 & otherwise \end{cases}$$

s.t.

where V_g = three-dimensional grid cube volume

(x_{gu}, y_{gu}, z_{gu}) = grid cube upper limit

(x_{gl}, y_{gl}, z_{gl}) = grid cube lower limit

The integration of measurement sensors on each agent within a team can cause the repetition of observational data. There is a need for faster assessment of sensory data. Thus, repetitive real-time sensory data are redundant and a liability. Figure 4.6 shows the progression of multi-agents across an indoor environment. It is clearly visible that in comparison to the agents' paths in the first image, the secondary image shows trajectories that map the same areas of free space. The agents accumulate redundant sensory data and neglect to map areas within the other rooms.

Cost function, ς_{12} encourages the removal of trajectories that fly across the same grid blocks. This is performed through the detection of overlapping flight areas. Each grid block is given the value of zero when ranking each path combination. It then has a value of one if there is an overlap.

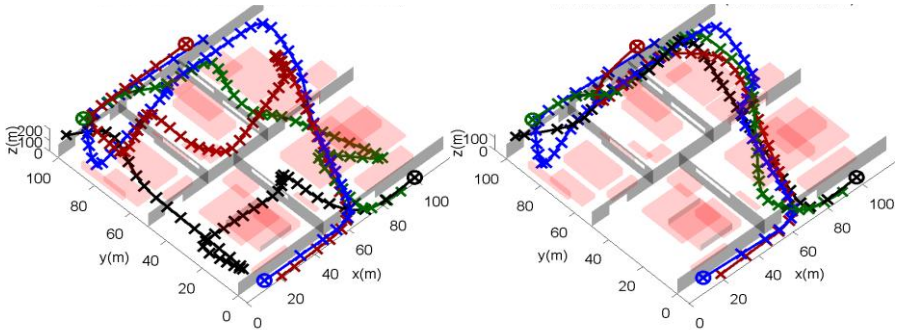


Fig. 4.6. Multi-agent low and high sensory data replication across indoor environment.

$$\varsigma_{12} = \sum_{1}^{A} \int_{t_{init}}^{t_{goal}} b_{\varsigma 12} \tag{4.21}$$

s.t.

$$b_{\varsigma 12} = \begin{cases} 1 & if \quad (x_{gl}, y_{gl}, z_{gl}) < (x_i, y_i, z_i) \ and \ (x_j, y_j, z_j) < (x_{gu}, y_{gu}, z_{gu}), \ \forall i = \{1,...N_A\}, \ \forall j = \{1,...N_A\} \\ 0 & otherwise \end{cases}$$

The 12 objectives that have been defined will be applied within the many-objective optimization process. These standardized MA-Spread objective functions are applicable to any number of agents as well as within any type of environment.

4.3. MULTI-AGENT QUADROTOR UAVS IN FORMATION FLIGHT

Along with Spread flight scenarios, there is a lot of recent interest in multi-robot motion in formation. There are many examples of flight in formation within nature. History shows us that many animals move and hunt in packs. Birds often fly as a collective flock. Insects also work as a team to transport food across far distances. The reason for this is

that a collective group is stronger and more effective than a singular individual. Formation flights are crucial to applications such as payload transportation or military missions such as security patrols. It is also often used for search and rescue missions at hazardous disaster sites. The agents in formation are required to perform cooperative sensory angular coverage and aerial flights whilst maintaining precise patterns. A group of agents that fly close together can perform many tasks that spatially spread agents can't.

This study has chosen to expand the standardize platform to include formation flight. This is due to the simplicity of path planning for a swarm of agents that fly in a formation structure. The trajectory planning for all agents is often done through the expansion of a singular leading agent's trajectory. This means that the basic trajectory planning process is much simpler than the MA-Spread application. The complexity of formation planning exists in the planning of the formation structure. The level of difficulty further increases when studies aim to create formation structures that adapt to their environment. Thus, the MA-Formation mission requires a multi-level program in addition to the basic path planning algorithm. It is these additional stages that make the MA-Formation application tougher to design in comparison to the MA-Spread mission. Multi-agent flights in formation introduce the additional complexity of preserving designated formation shapes whilst navigating through waypoints within a trajectory. These agents must fly closely whilst avoiding probable inter-agent collisions and environmental hazards.

There are four stages that exists within a formation planning algorithm. Some works implement all subsections whereas some only experiment with certain areas of formation planning. The first level is the design of the formation's reference trajectory. This process can be executed with most path planning algorithms. The second level involves the accurate free space mapping of the test environment. The formation planning algorithm must be capable of defining the amount of free space that exists around each path node. This can be performed in low or high resolution. The higher the free space mapping resolution, the more adaptable the formation shaped will be. The third stage involves the planning of the formation structures across the trajectory. There are two varieties of structures that can be applied. The using either rigid or nonrigid formation shapes is dependent on the mission at hand. Lastly, the trajectories for all agents can be designed from the reference trajectory and its formation structures. In this study, all four stages are implemented in order to provide the end user with a high level of flexibility.

Missions typically undertaken by UAVs in formation are generalized as dynamic contour maintenance formation flight (MA-Formation). This research implements a standardized platform for trajectory generation, coordination of agents in formation and

an optimization algorithm. It is applicable towards any variation of real world multi-agent cooperative tasks.

4.3.1 FREE SPACE SURFACE EXTRACTION

The first stage of formation planning has been described in the previous chapters. The paths containing waypoints for the formation flight are generated by GA and used as input for the formation planner. These paths are transformed into minimal jerk smooth splines. The time-based trajectory nodes are than applied as the reference coordinates for the dynamic formation planner. The planning of the formation shapes from initial towards the defined goal position is essential for collision free flights. In this study, the formation shape for N_A agents are designed based on the obstacle free space between close range obstacles and the agents. This formation planning algorithm creates symmetrical formation shapes. This constraint forces the formation shapes to constantly expand and contract across the trajectory. Thus, the symmetrical shapes fully test the algorithm.

This section describes the second stage of the formation planning process which is defined in Figure 4.9. Here, the free space of the environment is accurately mapped. Initially, six planes are constructed around each obstacle as shown in Figure 4.7. these planes form a cube that defines the space that the obstacle resides. The size of each cube is dependent on the free space mapping resolution of the test environment. Higher resolution free space mapping is essential for environments where the obstacles are not separately defined such as the mountainous terrain. Here, the mapping resolution of the free space is defined by the amount of height changes within the terrain. In these test spaces, there will be a high amount of obstacle planes. The number of formation reference trajectories is defined by the GA's population size. In this study, the formation planning algorithm is designed to be highly adaptive to the environment. Thus, each waypoint within a formation reference trajectory will be used to define the formation shapes of the quadrotors. The end user has the option of maintaining this highly accurate formation planner. They can also reduce the complexity and level of accuracy by designing the formation shapes for every few waypoints.

After placing the planes around all obstacles, the formation planner begins to extract the free space surface around each path waypoint. The algorithm needs to identify obstacle planes that are within close range of the current waypoint. Initial experimentation applied a more simplistic approach where the middle point of each plane is considered for the waypoint-to-plane distance measurement. It was found that for

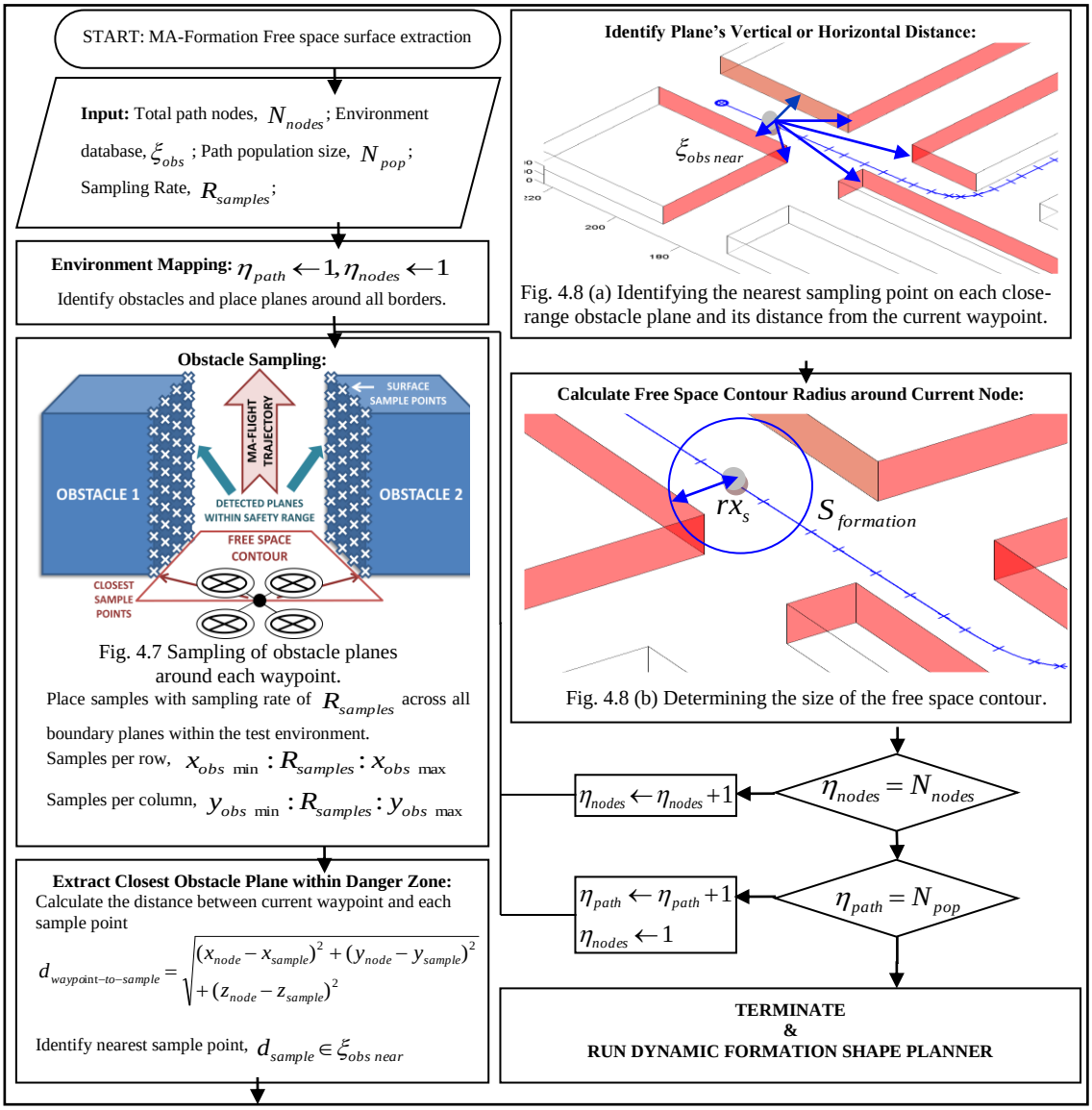


Fig. 4.7 Sampling of obstacle planes around each waypoint.

Fig. 4.8 (a) Identifying the nearest sampling point on each close-range obstacle plane and its distance from the current waypoint.

Fig. 4.8 (b) Determining the size of the free space contour.

Fig. 4.9 Flowchart of dynamic multi-agent formation planner.

planes that are longer in width or height, the centre point doesn't accurately define its distance from the waypoint. This is due to the fact that the corners of longer planes may be closer to the waypoint in comparison to the centre point. The plane may be neglected in the mapping of the free space contour because the distance of the centre point appears at a far distance. Thus, full sampling of all planes is applied instead. This produces more precise definition of the boundaries of the free space contour that will be relied on for the design of the formation shape at each waypoint.

Sample points are placed across the surface of these planes. The sampling rate is defined by $R_{samples}$. These sample points are used to detect the shortest distance between current waypoint, (x_c, y_c, z_c) and each plane. Firstly, the closest sample point for each

plane identified. This leads to the identification of the nearest plane for each waypoint. This close-range plane is stored within the database if it within the waypoint's danger zone. The planner then identifies if the close-range plane is parallel or perpendicular to the path. Then the radius of the free space surface around the waypoint is identified. This is done by defining the normal or tangential distances between the waypoint and the nearest plane as shown on Figure 4.8 (a). The vertical distance, d_{vert} is extracted if the plane is perpendicular to the path. On the other hand, the horizontal distance, d_{hori} is obtained if the plane is parallel to the path. This process produces a more precise definition of the free space surface's radius.

The surface as shown in Figure 4.8 (b) is mapped through the radius of free space at all angles from the formation waypoint. The depth of the surface is easily defined by the free space above and below the waypoint. This value can be obtained by evaluating the obstacle that are above and below the current waypoint. Then, this space between the obstacles defines the height of the free space surface, z_s.

The free space surface between the nearest obstacles and current waypoint is defined as,

$$S_{formation}(x, y, z) = [rx_s, ry_s, rz_s] \tag{4.22}$$

where $x_{obs}, y_{obs}, z_{obs}$ = nearest obstacle boundary planes

$[rx_x, rx_y, rx_z] = [x_{obs,} - x_R, y_{obs,} - y_R, z_{obs,} - z_R]$ = free space contour radius

The formation shape around each waypoint is designed based on the obstacle free space contour around the agents. The radius of the surface will determine if the quadrotors will be well spread or constricted to a small space whilst flying together. The accuracy of the formation planner is extremely important for tight spaces within the test environment. This is because there can be a risk for obstacle or agent-to-agent collisions across narrow passages.

4.3.2 FORMATION SHAPE PLANNER

The formation structure for the 8 quadrotors can be designed since the free space surface for each waypoint has been determined. The formation shape matrix for each agent, A is designed based on the free space surface's radius at each waypoint. The adaptive formation planner is capable of designing formation shapes for three possible cases. Firstly, the end user can define a default formation shape that can be used for formation flights across large spaces. This default shape allows the agents to maintain close-range flight despite the lack of near-range obstacles Next, a danger zone formation shape is

used for narrow passages and tight spaces. Lastly, the agents are spread across the free space surface in areas that aren't too wide or narrow. All three of these formation structures are shown in Figure 4.10. The agents are designed to always maintain a safe distance from their neighbouring agents. In this study, the sizes of the quadrotors within the images are exaggerated in order to clearly show the changes in formation shape. The end user has the flexibility of changing the minimum agent-to-agent distances and default formation shape.

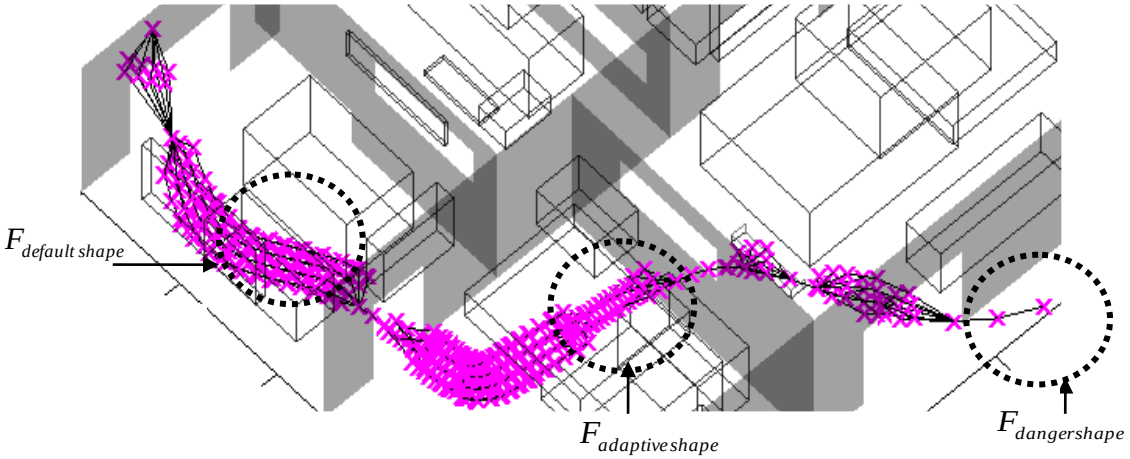


Fig. 4.10. Variations in formation shapes across a trajectory.

The first case is applied when there are no obstacles are within close sight. Here, a default formation shape, F_{shape} is applied. The default shape has the most number of agents per row in comparison to the other formation shapes. This shape is necessary to keep the agents in formation despite there being a large amount of free space around the agents. This allows them to quickly change into a smaller sized formation shape in the future. It is only applied whenever there are no obstacles within close range. In this study, a dual row formation shape is set as the default design. This divides the agents into two rows as seen in Figure 4.11 whilst considering the collision free distance, d_{ij} between agents.

$$F_{shape}(t) = \begin{Bmatrix} A_1 & \cdots & A_{N_A/2} \\ A_{(N_A/2)+1} & \cdots & A_{N_A} \end{Bmatrix} \tag{4.23}$$

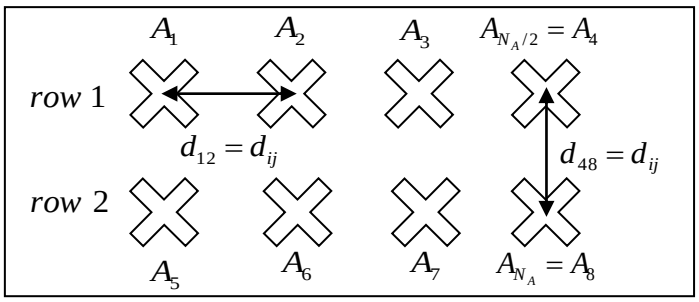


Fig. 4.11. Default formation shape.

The second case is applied for waypoints where the obstacles are within immediate range or extended narrow passages. This formation shape is defined through the free space contour between the waypoint and obstacles. The agents are spread across the obstacle free contour as seen in Figure 4.12. The number of agents per row is now defined as,

$$N_R = S_{formation}(x, y, z) / d_{ij}(x, y, z) \tag{4.24}$$

$$F_{shape}(t) = \begin{Bmatrix} A_1 & \cdots & A_{N_R} \\ A_{N_R+1} & \cdots & A_{2N_R} \\ & \vdots & \\ A_{(R*N_R)+1} & \cdots & A_{N_A} \end{Bmatrix} \tag{4.25}$$

where R = formation matrix row number

N_R = number of formation rows

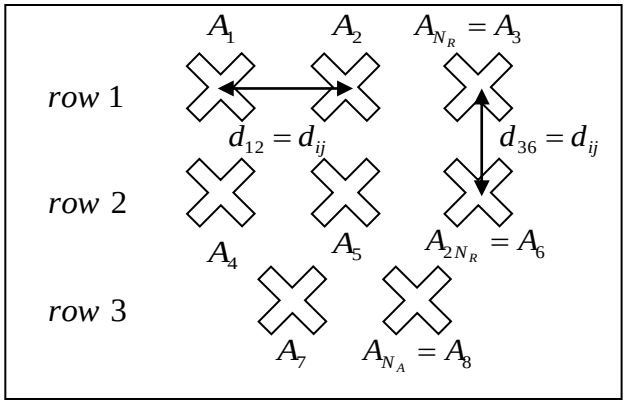


Fig. 4.12. Adaptive formation shape.

The last case is used when obstacles are within the danger zone. Thus, obstacles are within close proximity to the formation waypoints. Here, the formation shape is reduced to a one column or row matrix as seen in Figure 4.13. The formation can be guaranteed successful flight across the danger zone until reaching a vaster free space.

$$F_{shape}(t) = \begin{Bmatrix} A_1 \\ \vdots \\ A_{N_A} \end{Bmatrix} \tag{4.26}$$

The last level within the dynamic formation planner creates independent trajectories for each agent. This allows the closed loop controller to simulate the flight paths for each

agent individually. The predicted trajectories will let the optimization algorithm know if the formation trajectories and its formation shapes are feasible for real life flight.

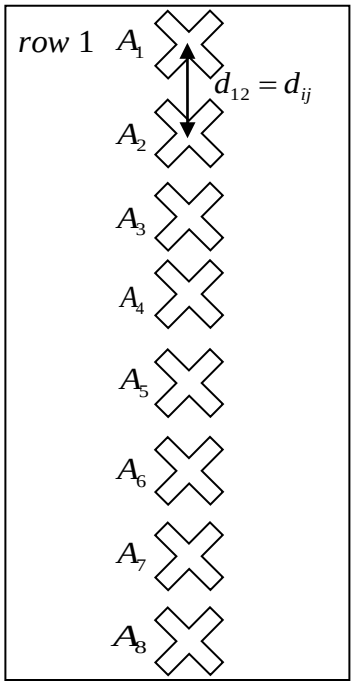


Fig. 4.13. Danger zone formation shape.

4.3.3 MULTI-AGENT FORMATION TRAJECTORIES

This section designs the flight trajectory for each quadrotor. The agents will be flying independently once they have received their desired coordinates. This process eliminates some of the coupling that exists in typical formation flight. This means that there is minimal coupling in the form of dependency between the movements between two agents. Each agent flies in their own direction whilst maintaining their formation shape. Thus, the agents will be able to fly collectively even if there is a faulty agent within their team. They will still be able to maintain their formation structure despite the removal of an agent or its communication link.

Figure 4.14 shows an example of how the agents are spread across the free space surface. The constant distance vector for each agent, $v_i = (x_{column}(A), y_{row}(A), z_{height}(A))$ is defined by the formation shape at each reference waypoint. Here, the formation reference trajectory's waypoints are used as the centre point for the formation structures. Firstly, the radius of the free space surface is used to define the amount of space that exist at the left and right of the centre point,

$$x_{left} = -[rx_s, ry_s] \tag{4.27}$$

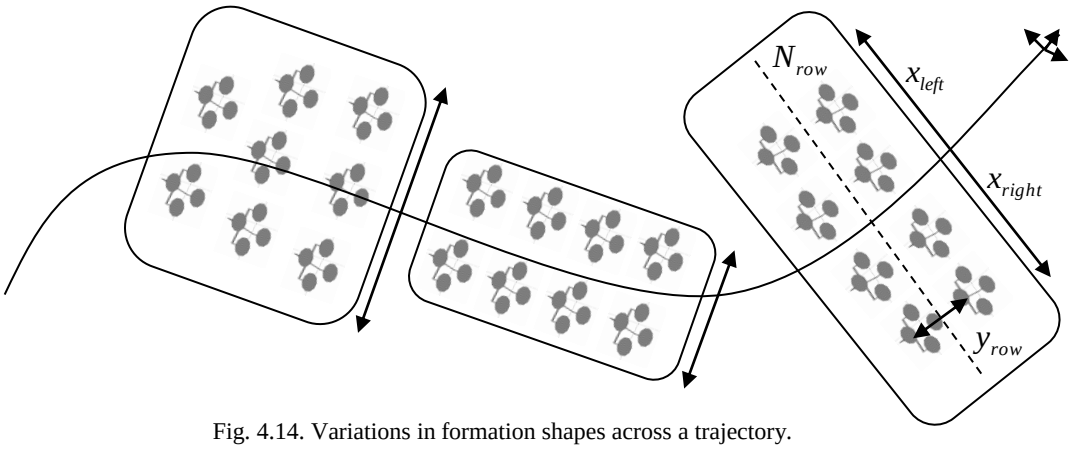


Fig. 4.14. Variations in formation shapes across a trajectory.

$$x_{right} = [rx_s, ry_s] \tag{4.28}$$

Next, the agents that are within the same row are divided across this space. The x-axis coordinates for each agent is defined by,

$$x_{column}(A) = x_{left} : d_{ij} : x_{right} \tag{4.29}$$

The y-axis coordinates for each agent is determined based on the row that it belongs to,

$$y_{row}(A) = -d_{ij}(N_{row} - 1) \tag{4.30}$$

Lastly, the z-axis coordinates for all agents, $z_{path}(A)$ is the same as the formation reference path's z-coordinate, $z_R(t)$. This means that the agents fly at the same altitude.

The flight trajectory of each agent, $\Im_A(t)$ is based on the reference trajectory's rotational and translational movements, $R(\psi)$. It is also dependent on the previously defined distance vector between the centre of the formation structure.

$$\Im_A(t) = \Im_R(t) + R(\psi)\ v_i \tag{4.31}$$

where, $R(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

The path waypoints for each agent are,

$$x_A(t) = x_R(t) + x_{column}(A)\cos(\psi) + y_{row}(A)\sin(\psi), \quad (4.32)$$

$$y_A(t) = y_R(t) - x_{column}(A)\sin(\psi) + y_{row}(A)\cos(\psi), \quad (4.33)$$

$$z_A(t) = z_R(t) \quad (4.34)$$

All three stages of the dynamic formation planner are completed with speed and accuracy. Both the multi-agent quadrotor trajectories and its formation designs are then applied towards determining the values of the many objective functions.

4.3.4 MA- FORMATION CONTROL SYSTEM

The control system for the MA-Formation application has the same controller and quadrotor's mathematical model as the MA-Spread mission. Similarly, most of the information flow across the entire trajectory optimization algorithm remains the same. There are some modifications to the control system that are designed specifically for multi-agent quadrotor formation flight. The MA-Formation mission has two planners as shown in Figure 4.15. There is one for trajectory planning and another for formation planning. The control system must test the feasibility of the data that is generated by both these planners. It has to determine if the agent's planned trajectory can be tracked. The estimator must also determine if all the agents can collectively track the designed formation shape as well.

In this research, eight quadrotors, $i = 1:8$ are flown in formation across different terrains. Firstly, the control system requires predetermined input parameters for the default formation shape, $F_{default}$ as well as the minimum agent-to-agent distances, $d_{ij\min}$. The MA-RRF and GA path planners are only used to plan the reference trajectory for the formation flight. Thus, the path planning process is much faster than the MA-Spread mission. It is extremely important that the control system is able to predict the flight path of each agent. This means that each agent must be provided with its own trajectory. Here, a formation planner is used to design the trajectory of an individual agent, $\chi_i(t)$. The formation planner extracts the free space contour, $S_{formation}(x, y, z)$ that exists around each waypoint in the reference trajectory, $\chi_{reference}(t)$. Next, the planner defines the formation shapes, $F_{shape}(t)$ that fit within the free space contours. Lastly, the trajectories for each agent is designed based on the formation shapes across the reference path. This process creates a decentralized system where each agent flies independently. Thus, the control system for each agent isn't dependent on its neighbouring agents and can be run in parallel. Each controller requires the same input data. As with the previous application, the estimator requires the desired positional coordinates and rotational angles. It also needs the initial values for the quadrotor's position and its derivatives.

As with the MA-Spread mission, the output of the control system will determine the values of the many objective functions. The MA-Formation task has some of the same costs as the prior independent flight application. The parallel run control system produces the predicted positional and rotational data for the team. These values are obtained from the quadrotor's mathematical model that also provides information on an agent's velocity and acceleration. This information is compared with the planned trajectories. It is an indicator that the agents are not able to maintain their formation structure when their path deviation error is high. The flight time is also predicted through the collective flight time of all eight agents. The objective functions that are directly related to formation flights require predicted data such as the number of formation shape changes, F_{shape} . It also relies on the estimated formation rise time, t_{rise} of each agent. The rise time that is obtained by the controller is the time it takes for each agent to morph from its previous formation structure, t_{before} into a new shape, t_{after} . This data shows the complexity of the formation shape changes across each trajectory. Lastly, the estimated distance between neighbouring agents, d_{ij} is highly important for formation flight. This allows the optimization algorithm to determine if the agents are capable of maintaining their relative positions whilst flying.

The predicted flight paths of all eight agents will determine if the reference trajectories and formation shapes are feasible for real life flight. Formation reference trajectories that are not feasible will not be maintained within the next generation's population. The many objectives optimization algorithm will rank less complex formation trajectories above those with complicated shape changes. The DRMOO ranking system will be described in further detail within the end of this chapter.

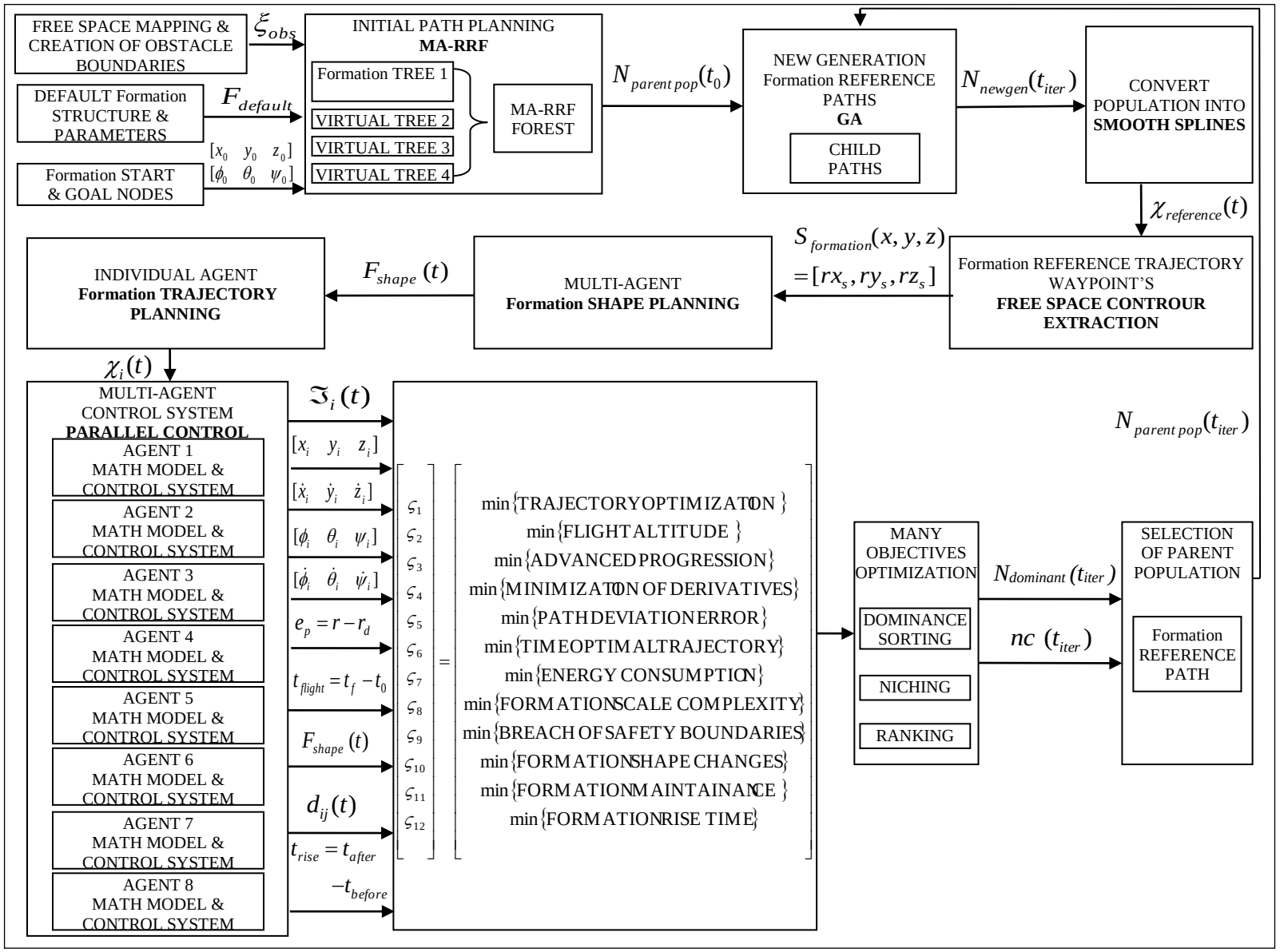


Fig. 4.15. MA-Formation closed-loop nested control system.

4.3.5. MA- FORMATION OBJECTIVE FUNCTIONS

In addition to the standardized objective functions, four objective functions specifically catered to the application of formation flight scenarios is shown below. The data from the flight predictions of all agents is used to determine the values of the 12 objectives. The parameters applied within these objectives are shown in Table 4.2 [141-142]. These objective functions are designed to mimic the costs and constrains faced by multi-agent quadrotor UAVs in real life formation flights.

TABLE 4.2: MA-Formation PARAMETERS

Description	Value	Description	Value
Population size	30	Obstacles Plane Detection Range	20 *m*
Selection Rate	0.5	Default Number of Agents/Row	4
Safety Zone Obstacles Boundary	$\xi_{obs} \pm 6$m	Default Number of Column/Formation	2
Number of Gaui 330X-S Agents	4	Agent-to-Agent Minimum and Maximum Distance	[2, 3] *m*
Gaui 330X-S Agents Maximum Fuel	15mins	Formation Minimum and Maximum Maintenance Distance	[1.5, 3.5] *m*
Number of Fyetech Agents	4	Redundancy Similarity Threshold	32.5%
Fyetech Agents Maximum Fuel	10mins	Obstacles Plane Detection Range	20 *m*
$R_{samples}$[city, indoor, mount]	[5,2.5,4] *m*		

The next cost function minimizes the number of formation shape changes within a trajectory. Firstly, this cost reduces the complexity of the formation flight mission by choosing paths with minimal shape changes. Secondly, it allows the end user to plan for load lifting. This cost function is extremely significant for the transportation of payloads through a team of quadrotors that requires the attachment of the load to the agents be maintained throughout flight. The first image of Figure 4.16 shows a formation trajectory that requires shape contractions and expansions. Whereas, the second maintains the same shape which is necessary for load lifting. If the objective value is $\varsigma_{13} \neq 0$, this trajectory is rejected for payload transportation. Here, the rigidity of the formation is imperative for successful transportation.

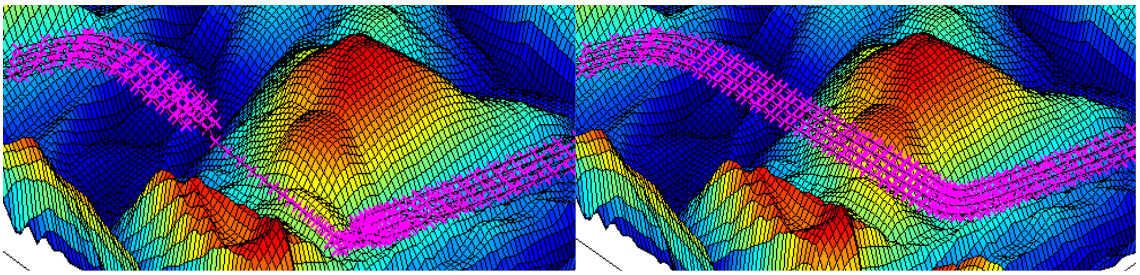


Fig. 4.16. Trajectories of nonrigid and rigid formation shapes across mountainous terrain.

$$\varsigma_{13} = \int_{t_{init+1}}^{t_{goal}} b_{\varsigma 13}(t)\ dt \tag{4.35}$$

s.t. $$b_{\varsigma 13} = \begin{cases} 1\ , if\ F_{shape}(t-1) \neq F_{shape}(t) \\ 0\ , otherwise \end{cases}$$

where $F_{shape}(t)$ = formation shape matrix

Formation design conservation requires the cooperation of all agents within its team to maintain designated relative distance between neighbouring agents. There may be a break is formation if the designed trajectory requires the agents to manoeuvre aggressively around sharp bends. Similarly, if the contraction and expansion of formation shapes are too extreme for the allocated time, the formation design becomes unattainable in real life flights. If trajectories that are infeasible for formation maintenance are detected, the optimization process will gradually reduce its occurrence within its population.

Cost function, ς_{14} tests if the agents are retaining their formation shape at each sample time. In this case, the estimated formation flight is compared to the formation shape designed by the planner. Trajectories with hard to follow shapes are slowly filtered out of the population.

$$\varsigma_{14} = \sum_{1}^{A} \int_{t_{init}}^{t_{goal}} b_{\varsigma 14}(t)\ dt,\ \ \forall A = \left\{ 1,...N_A \right\},\ f_{ij,\min} \leq \Delta f \leq f_{ij,\max} \tag{4.36}$$

s.t. $$b_{\varsigma 14} = \begin{cases} 1\ , if\ \Im_A(t) \neq f_A(t) + \Delta f \\ 0\ , otherwise \end{cases}$$

where $f_A(t)$ = agent's position relative to neighbouring agents

Δf = distance between neighbouring agents

f_{ij} = formation maintenance buffer range between neighbouring agents

The complexity of formation changes within a trajectory is measured through the scale of shape expansion or contraction required. This cost function is applied to determine the ratio of between two formation shape changes.

$$\varsigma_{15} = \sum_{1}^{A} \int_{t_{init}}^{t_{goal}-1} b_{\varsigma 15}(t)\ dt,\ \ \forall A = \left\{ 1,...N_A \right\} \tag{4.37}$$

s.t. $$b_{\varsigma 15} = \begin{cases} N_{col}(t) / N_{col}(t-1), if\ F_{shape}(t) \neq F_{shape}(t-1) \\ 0\ , otherwise \end{cases}$$

Cost function, ς_{16} measures the percentage of total flight time that is dedicated into changing formation shapes. It can tell the user if the changes in formation shapes across the current path are complex and require more time in comparison to the shapes across another formation path. Thus, the user can identify if the longer flight time is due to the formation shape planning or the formation path length itself. As shown in Figure 4.17, the rise time is derived from its previous formation shape, t_{before} to the period where all agents have merged into the next formation shape, t_{after}. These values are obtained from the control system. Thus, this cost function measures the percentage of total flight time if the formation that is dedicated to morphing into changing formation contours.

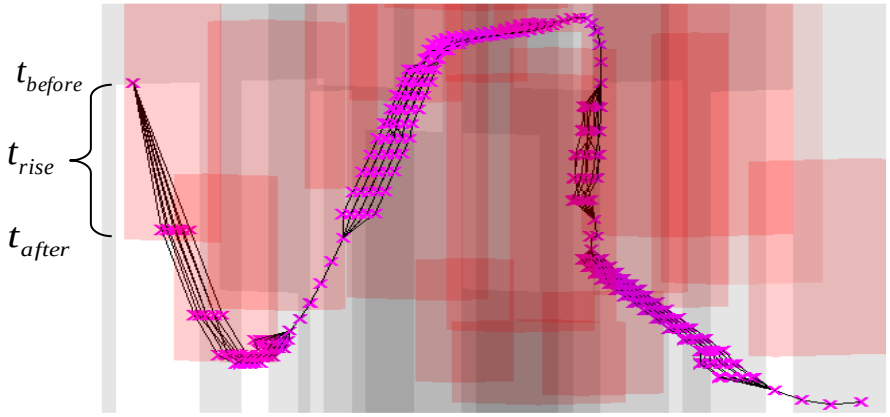


Fig. 4.17. Rise time between previous formation to the next formation shape across indoor environment.

$$\varsigma_{16} = \int_{t_{init}}^{t_{goal}} t_{rise} = t_{after} - t_{before}\ dt \tag{4.38}$$

s.t.
$$t_{before} = \begin{cases} (t-1), if\ \Im_A(t-1) = f_A(t-1) + \Delta f \\ 0,\ otherwise \end{cases},\ \forall A = \{ 1,\ldots N_A \}$$

$$t_{after} = \begin{cases} t, if\ \Im_A(t) = f_A(t) + \Delta f \\ 0,\ otherwise \end{cases},\ \forall A = \{ 1,\ldots N_A \}$$

The values of these 12 objective functions will be used to rank and sort the formation trajectories across each generation. The ranking process will be performed by a many-objectives optimization algorithm. In this thesis, the DRMOO algorithm is utilized to perform the sorting of the multi-agent trajectories for both the MA-Spread and MA-Formation missions.

4.4. LARGE DIMENSIONAL MANY-OBJECTIVES OPTIMIZATION

When attempting to plan trajectories for various real-world missions, the complexity of processing high-dimensional becomes apparent. In this study, the difficulties and challenges of the simultaneous optimization of many objectives for multi-agent quadrotors within different terrains is presented. The target for applying many-objective optimization towards trajectory generation for multi-agent quadrotors is to provide a diverse yet well minimized solution set to users. When trying to optimize many objectives, it promotes an understanding that there is never just one optimal solution that is best in regard to all cost functions. This study also aims to remove the need for prior determination of the possibility of optimizing many objectives as opposed to just a few. This platform allows the user to evaluate the pros and cons of various criterions through visual three-dimensional environment mapping as well as data driven analysis when deciding which trajectories to implement for real-time flight. Thus, the end user is provided numerical value for the trade-off variations in cost values for each solution within the set of optimized trajectories.

The initial suboptimal trajectories developed by the MA-RRF algorithm for all three test environments are applied as input for the optimization process. Using Genetic Algorithm, these trajectories are meshed to create new paths through the crossover and mutation process. Next, the control system prior produced is applied to generate predictions of the cost values of all objectives. Based on these estimations, the process of optimization is run to rank and filter suboptimal and non-diverse trajectories. Here, we evaluate the capabilities of the hybridized Dimensionality Reduced Many Objectives Optimization Algorithm to produce a batch of well minimized trajectories where all objectives are considered to be equally essential. These trajectories are with various strengths and weaknesses for these differing terrains. Through these experiments, it is shown that the algorithm can be implemented as a standard platform for any multi-agent scenario or application.

4.4.1 MULTI-OBJECTIVES OPTIMIZATION

Multi-objectives optimization is often applied towards real life problems that require a well minimized solution. Humans perform simple optimization processes in their daily lives. Most people are required to transport themselves to their workplaces daily. It is important that a person picks the optimal form of transportation. An error in judgement can lead to the individual being extremely late to work. In this case, the possible transportation options are a car, bicycle, motorcycle, bus or train. The multiple objectives that are involved in this optimization problem are the total cost, travel time, comfort and walking distance. All objectives are of equal importance. Some of these objectives will

also conflict with each other. The individual may also have constraints that need to be considered such as a monthly budget or office hours.

The multi-objective optimization problem can be defined by:

Minimize/Maximize M objective functions: $f_m(\mathbf{X}),\ \ m = 1,2,\ldots, M_i$ (4.39)

Subject to: J Inequality constraints $g_j(\mathbf{X}) \geq 0,\ \ j = 1,2,\ldots, J_i$ (4.40)

K Equality constraints $h_k(\mathbf{X}) = 0,\ \ k = 1,2,\ldots, K_i$ (4.41)

Upper and lower bounds $x_i^{(L)} \leq x_i \leq x_i^{(U)},\ \ i = 1,2,\ldots, n$

where solution X is a vector of n decision variables: $\mathbf{X} = (x_1, x_2, \ldots, x_n)^T$

There are two popular methods for applying multi-objective optimization. The two options differ based on when the end user's preferences are applied within the optimization process. The first option is the preference-based multi-objective optimization method. This option is often used by many researchers because of its simplicity in application. There are some optimization problems that can be solved with the end user's preferences in mind. These are problems where the end user knows the level of importance of each objective. These preferences will be used to set the weight of each objective within the aggregate cost function. Here, a composite objective function, f_{total} is formed from the multiple objectives.

$$f_{total} = \sum_{m=1}^{M_i} \omega_m f_m(\mathbf{X}),\ \ \omega_m = [0,1] \tag{4.42}$$

The preferences of the end user are defined through the weights, ω_m that are attached to each objective. The value of each weight is proportional to the importance of its objective function. In this case, the end user is supplied with a singular optimal solution at the end of the optimization process. There are some studies that aim to find a collection of solutions by applying a variety of weight values. This process will require the intuition of an individual that is experienced with the objectives within the mission's optimization problem.

A less subjective method would be the optimization of all objectives equally without any prior preferences. This method more closely mimics real life optimization problems. It also removes the bias that can push the algorithm to minimize certain objectives only. Firstly, all the possible solutions are considered. The objective values for each solution is evaluated. Then, all solutions are compared to each other. This process determined which option provides better trade-off values for all objectives. The options

that contain well minimized cost functions are maintained within the solution set. This procedure continues until there are no better solutions. Finally, the end user received a detailed breakdown of the objective values for each solution. It offers the end user information about the amount of optimization that is possible for each objective.

Each transportation option in the previously discussed problem produces different values for each of the objective function. Thus, each option has different trade-offs in terms of advantages and disadvantages. For example, driving a car may be comfortable but it comes at the expense of paying for more fuel. On the other hand, taking a train means lower levels of comfort but the tickets are cheaper. An individual needs to know the trade-off of each option in order to make a knowledgeable decision. Thus, optimization is defined as a tool for finding and comparing different solutions. It is important to understand that since there are multiple conflicting objectives that are being optimized, there can never be one solution that optimizes all objectives. There are many good solutions with different trade-offs. The final solution set will be made up of multiple optimal solutions that are defined based on their objective values. The individual gets to compare these good solutions and select the best transportation method based on their personal preferences. Thus, it is advantageous that the user doesn't need to define prior preferences. They have the flexibility to evaluate these optimal solutions post optimization.

In this research, the secondary method of multi-objective optimization is applied. No prior preferences are applied within this study. Multi-objective optimization algorithms require a large population of feasible solutions. A large population allows the algorithm to retain a variety of solutions with different objective values across the generations. It also increases the solution search space and reduces the chances of premature termination before finding well minimized solutions. This work aims to produce a diverse population of well minimized trajectories for multi-agent quadrotors. This means that the end user is supplied with a large collection of options to compare and choose from.

Many researchers have shown that evolutionary algorithms are ideal for multi-objective optimization. Algorithms such as GA are capable of generating a large collection of multi-agent quadrotor paths as an input. Similarly, it also produces a large amount of multi-agent quadrotor paths as an output. GA also allows the end user to introduce elitism during the selection process at each generation. Thus, the optimization algorithm is able to maintain good solutions within the population whilst searching for new ones.

GA alone isn't suitable for many-objectives optimization because its simplistic ranking process can't identify which solutions are better than the other in regard to all twelve objectives. It requires a more refined algorithm to perform the sorting process at each generation. This is explained in more detail in Section 4.2.2-4.5.3. Here, a variation of MOO, many-objective optimization algorithm will assist the GA in sorting and ranking these solutions. This is performed by maintaining dominant and diverse solutions. The targets of a many-objectives optimization algorithm can be defined as:

1. To find a collection of solutions that is optimal.
2. To maintain a diverse set of optimal solutions.

These two targets allow the optimization algorithm to preserve solutions with multiple trade-offs across generations. Both targets are often conflicting with each other. In many cases, an optimal solution can be similar to other solutions within the population. Likewise, a solution with different trade-offs in terms of cost values may not be an optimal solution. All optimization algorithms have a tough challenge of creating a balance between both goals.

4.4.2 WELL MINIMIZED SET OF SOLUTIONS

As previously defined, the first objective of a multi-objective optimization algorithm is to identify well minimized or dominant solutions. Pareto-optimal solutions are a popular term for a set of dominant solutions. This term originates form the Pareto-optimal front. The Pareto frontier is defined as the curvature obtained when all the Pareto optimal solutions are joint together and viewed as a whole as shown in Figure 4.18. The Pareto frontier plays an important role in the implementation of multi-objectives optimization. In a situation where a large number of solutions are being considered, some method for ranking them must be applied to reduce weaker solutions. The approximation of the multidimensional Pareto frontier is determined through the classifications of solutions. The designer of a MOO algorithm must be able to identify these optimal solutions and understand the importance of finding them.

The end user applies a MOO algorithm to find solutions to a problem that has many costs. This means that the optimization algorithm must place equal importance to all objectives. The end goal of the algorithm is to produce a set of solutions that minimize all costs simultaneously. These solutions are called the dominant or nondominated Pareto-optimal solution set. Firstly, any member of the Pareto-optimal set dominates other solutions that are not within the set. Thus, this optimal solution has better costs values than the other non-optimal solutions in regard to all objectives. Secondly, no solution within the dominant set can be said to be better than the other with respect to all

objectives. In this case, solutions within the dominant set will produce different variations of minimal cost values. They will be better than each other regarding some objectives only.

These rules allow the algorithm to identify if a solution is optimal. A solution f_{1A} is said to dominate solution f_{1B} if both conditions are met:

1. Solution f_{1A} is no worse than f_{1B} in terms of all objectives.
2. Solution f_{1A} is strictly better than f_{1B} in at least one objective.

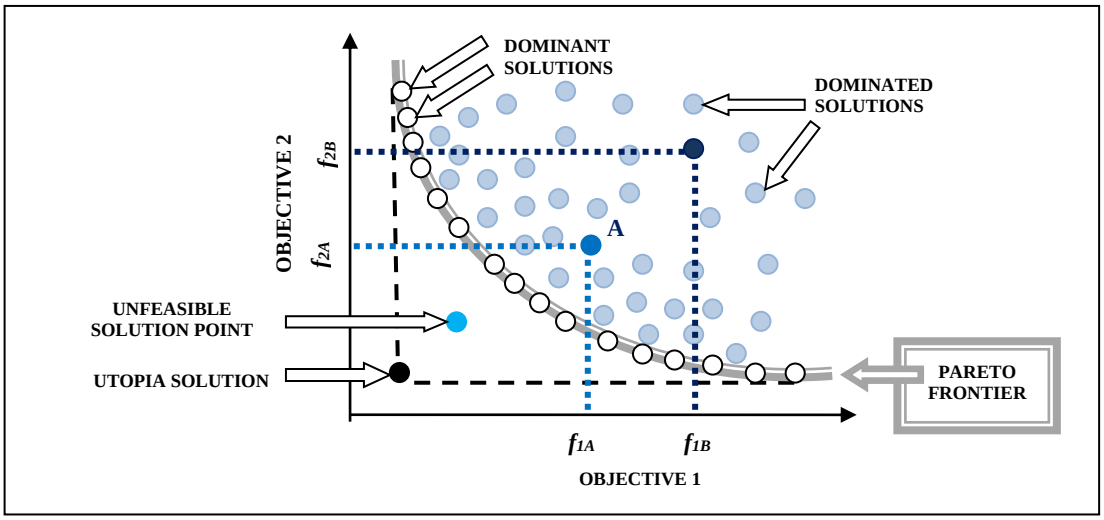


Fig. 4.18. Optimal and suboptimal solutions across the Pareto frontier.

Here, solution A, f_{1A} is defined as dominating solution B, f_{1B} when both conditions are fulfilled. Firstly, solution B is dominated by A. Figure 4.18 shows that both of the objective values for solution A are better than B. Secondly, solution A is non-dominated by B. It can be seen that solution B doesn't have any objective values that are more minimized than A. Lastly, solution A is non-inferior to B. Figure 4.18 shows two solutions A and B are located across different areas of the graph. Initially, solution A is classified as a nondominant solution and B as a dominated solution. As the iterations progress, there will be other solutions that dominate solution A as well. Slowly, the dominant solution set will slowly converge towards the Pareto optimal frontier. This will be accomplished through the constant comparison and maintenance of nondominated solutions.

There are many studies that are capable of identifying the Pareto-optimal solution set. It can be tougher to obtain the Pareto frontier with more complex problems. Some works choose to focus on converging towards the Pareto frontier instead. This way, an

approximation of the Pareto optimal solutions is performed. The ranking process that is applied can be highly advantageous for real life problems even though the Pareto-optimal solution set isn't identified. It is a tool to remove suboptimal solutions and maintain the best solutions within a population. It still offers the end user a collection of solutions that offer minimal costs and various trade-offs.

4.4.3 DIVERSE SET OF SOLUTIONS

The second goal of a multi-objective optimization algorithm is to maintain a diverse set of well minimized solutions. Typical diversity mechanisms such as the mutation operator may not be sufficient for maintaining a variety of solutions. In many cases, additional means of diversity management must be implemented when dealing with multiple objectives.

There are different varieties of diversity management solutions. In this research, niching is used to identify solutions that are diverse. In biology, an environmental niche is a term that is used to define locations that contain organisms that cater specially to a particular species. Similarly, a niche market is a business term that identifies consumers that enjoy similar products. The common theme in all these definitions is that a niche describes a group of similar organisms or behaviours. The members of a niche collectively form a cluster. These concepts can also be used for diversity management within mathematical problems that have multiple objectives. The term niche is now applied within many MOO algorithms to group together solutions that have similar objective values.

It is essential that an MOO algorithm produce a good Spread of various trade-off well minimized solutions amongst different objectives. The representatives of all solution optima are necessary to allow higher-level information to select the best solution. Working with multiple objectives produces many optimum solutions in the form of global and local optima as shown in Figure 4.19. Here, the algorithm gives priority towards the adequate representation and maintenance of the many local and global optima through generations. If the number of solutions within a cluster are too large, its optima is over represented within the population whereas, is the cluster size is too small, the optima becomes under represented. The algorithm must strike a balance between maintaining similar clusters of well minimized solutions at the risk of the removal of diverse clusters of suboptimal solutions.

In this study, a niche function is used to define the spatial distribution of the solutions within the high dimensional space. There are three variables within the niche function. The first is the Euclidean distance between two solutions, d_{ij} . Next, a sharing

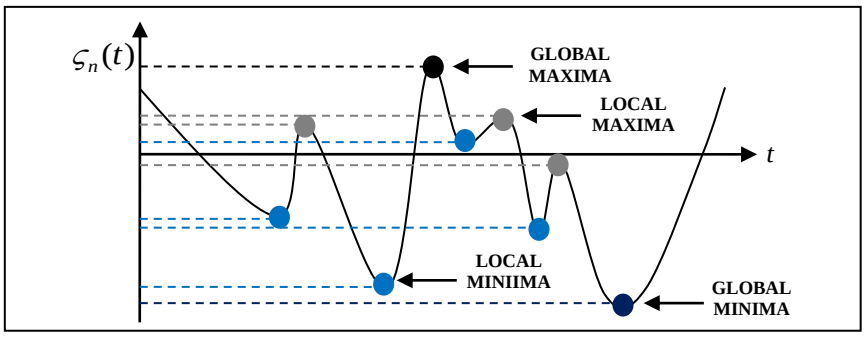


Fig. 4.19. Global and local optima of different objectives.

function, $Sh(d_{ij})$ is used to define the level of similarity between two solutions. The third variable is the niche size, σ_{share} which is shown in Figure 4.20. This variable defines the radius of similarity between solutions within objective space. Two solutions that are at a distance that is less than the niche size is considered to be a part of the same cluster. Thus, value of the niche size dictates the probability of detecting a higher or lower number of optima.

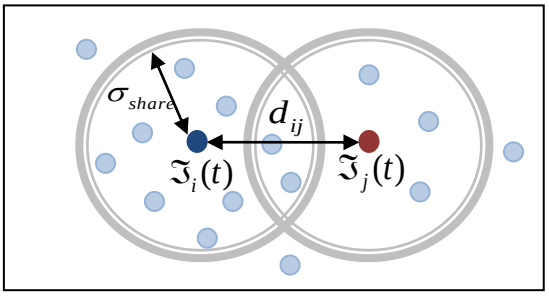


Fig. 4.20. Adaptive niche radius within of clusters of solutions.

For each solution, $j = 1, ..., N_{sol}$ the distance with all other solutions, $i = 1, ..., N_{sol}$ is calculated through,

$$d_{ij} = \sqrt{\sum_{1}^{n} \left(\varsigma_n{}^i - \varsigma_n{}^j\right) / \left(\varsigma_{n,\max} - \varsigma_{n,\min}\right)}, \quad \forall n = \left\{ 1, ..., N_f \right\} \tag{4.43}$$

where N_f = number of objectives that need to be minimized

The sharing function, $Sh(d_{ij})$ creates the comparison between two solutions as to sharing of each optimum.

$$Sh\ (d_{ij}) = \begin{cases} 1 - d_{ij} / \sigma_{share}\ ,\ if\ d_{ij} \leq\ \sigma_{share} \\ 0\ ,\ otherwise \end{cases} \tag{4.44}$$

The summation of the sharing function defines the niche count, nc_i for each solution is then used as a measure of the percentage of the total solutions that belong to a certain optimum. The application of niche count defines if a solution's niche is crowded. It determines how many solutions are within a solution's niche radius. It encourages the degradation of crowded solutions and the enhancement of cluster representative solutions.

$$nc_i = \sum Sh\ (d_{ij}) \tag{4.45}$$

As previously defined, this research combines a number of diversity management systems such as path mutation, similarity filtering and niching. This gives the algorithm the best chance at producing and keeping highly diverse Pareto-optimal solutions throughout the first to final generations.

4.5 DIMENSIONALITY REDUCED MANY-OBJECTIVES OPTIMIZATION

As initially described, this planning and optimization algorithm aims to generate a large collection of trajectories for multi-agent quadrotors. This section describes the DRMOO algorithm in detail. There are two main components within the DRMOO algorithm. The first subsection is the ranking of solutions through the creation of objective subsets. This study utilizes dimensionality reduction in order to generate smaller objective subsets [133,136]. The algorithm begins with the full objective set and creates objective subsets that hold no less than three cost functions. The full objective set and its subsets are used in rotation. Thus, the algorithm is able to perform both local and global optimization simultaneously. Whilst this process does mimic multi-objective optimization, the full set of objectives is still maintained within the algorithm.

The secondary subsection of the algorithm involves the maintenance of diverse trajectories within the solution population. This study applies adaptive niching in order to identify clusters of similar solutions [139]. The adaptive niche radius changes across generations based on the average distance between neighbouring solutions. This adaptive diversity operator allows the algorithm to effectively identify local optima within the search space. Each cluster of solutions only requires a small number of representative solutions. The niching process penalizes solutions that are within crowded clusters. Similar solutions within these crowded clusters are slowly removed from the population

as the generations' progress. This way only the representative solutions of each local optima is maintained.

The combination of dimensionality reduction and adaptive niching will transform the typical multi-objective optimization into an effective many-objectives optimization algorithm. The end user will be able to optimize many objectives within the multi-agent quadrotor trajectory planning algorithm.

4.5.1 MANY-OBJECTIVES OPTIMIZATION

Many-objectives optimization is an extension of multi-objective optimization. There is one key difference between the two varieties of optimization algorithms. The term many-objectives optimization is typically dedicated to the optimization of more than three objectives, $\varsigma(t)$ simultaneously that are often conflicting in nature. In this study, the values of all objectives are minimized across generations. Any objective that needs to be maximized can be converted to a minimization problem by multiplying the equation with -1. The many-objective optimization problem is defined as,

$$\min[\varsigma(t)] = \min[\varsigma_1(t),\ \varsigma_2(t),\ \varsigma_3(t),\ .....,\ \varsigma_n(t)]\ ,\forall n = \{\ 1,\ ...,\ N_f\ \} \quad (4.46)$$

where N_f = number of objectives

Most real-life issues can be solved in many ways. It is very rare that a problem only has one possible solution. Similarly, it is important to consider many costs before choosing a solution. Each solution will have its own advantages and disadvantages. This algorithm is designed with an understanding that there is never just one optimal solution that is best in regard to all cost functions. Whilst many studies choose to simplify and prioritize certain cost functions, real life flights require the consideration of many objectives simultaneously. As previously defined, there are works that do consider many objectives through an aggregated weighted cost function. The disadvantage of using weights is it creates bias during the optimizations process. Similarly, it also requires the end user to provide predetermined weight values. This can be challenging because the end user doesn't have prior knowledge of how much each cost function can be minimized.

Another way to solve a problem that has many costs is through multi-objectives optimization. The concepts that are applied within multi-objective optimization algorithms are also used within many-objective optimization problems. The solutions within a population can be ranked by comparing their level of dominance and diversity. One important factor to consider when applying these concepts to many-objective optimization is that it is a much more complex problem. The number of objective functions is higher in comparison to a basic multi-objective optimization problem. Thus,

some changes must be introduced when applying these concepts towards many objectives.

The first challenge that occurs when applying many objectives within a typical multi-objectives optimization algorithm is that the selection pressure is greatly reduced. The optimization process involves two sections which are the ranking of well minimized solutions and the maintenance of diverse solutions at each generation. The ranking of solutions is highly challenging when dealing with many objectives. This occurs because the selection pressure towards optimal solutions are reduced due to the extensive number of dominant solutions within the population. Most solutions are labelled as dominant because it is difficult to dominate a solution in regard to all objectives. Thus, the algorithm is incapable of sorting the solutions based on their level of dominance. In this case, priority is skewed towards obtaining diverse solutions that may not be part of the optimal solution set. There is a crucial need for an additional mechanism to drive the optimization process towards maintaining both optimality and diversity.

The next challenge that occurs within a many objectives optimization algorithm is the visualization of the Pareto frontier. In most multi-objective optimization studies, the authors are able to present proof of convergence. The images presented will show that the solutions within the final population have converged towards the Pareto front. This provides reassurance that the final solutions are indeed optimal. The image will also show the level of diversity within the final population. The Pareto frontier allows the end user to measure the effectiveness of the optimization algorithm. The visualization of the Pareto frontier within a many objectives optimization is extremely difficult. In some cases, it can be impossible. This is because the dimension of the frontier is dependent on the number of objectives that is implemented. A typical multi-objectives problem uses three or less objectives. This means that the Pareto frontier is a two or three-dimensional image. On the other hand, the Pareto front becomes a high dimensional image when many objectives are applied. Proof of convergence can be challenging when the Pareto front cannot be visualized.

Here, we apply dimensionality reduction towards increasing selection pressure without the absolute removal of any objectives from the many-objectives optimization process. The algorithm optimizes the population the best it can until a termination point is obtained. It prioritizes the maintenance of well minimized solutions within the population as opposed to obtaining the Pareto optimal front. Thus, the solutions that are presented to the end user are well minimized and diverse.

4.5.2 DIMENSIONALITY REDUCTION

In this study, a hybrid Pareto ranking algorithm that combines dimensionality reduction and partial Pareto dominance is applied. Initially, dimensionality reduction is applied to reduce the number of objectives that are optimized at one time. This process creates smaller groups of objective subsets from the full many objective set. Then, Partial Pareto dominance is utilized to rotate the smaller objective sets across generations [136]. This algorithm is used to increase selection pressure without the absolute removal of any objectives from the many-objectives optimization process. A flowchart that describes the optimization algorithm is presented in Figure 4.21.

This algorithm is based on the level of conflict that exists between a collection of objective functions. It aims to approximate the amount of dependency between two objectives. The GA designed trajectories are used to estimate the values of each objective function. There are two types of relationships that can occur between two objectives. In the first case, the progression of two objective functions may show that the values of one objective increases with time whereas the other decreases. This shows that the two objectives are in conflict with one another. There is a low level of coupling or dependency between these two objectives. This is because the minimization of one objective will cause the maximization of the other objective. It is important to maintain both objectives within the optimization process since they must be independently minimized. In the second case, the values of two objectives increase or decrease together. This means that there is minimal conflict between the two objective functions. One can assume that if two objectives are minimized simultaneously, the algorithm only needs to consider one objective to optimize the other. This renders the nonconflicting objective redundant within the optimization process. The identification of redundant or nonconflicting objectives can be useful for a many-objectives optimization algorithm. The redundant objectives can be removed from the objective set without causing many changes to the ranking of the solutions. Thus, it allows the algorithm to increase selection pressure by comparing only a few objectives at a time.

There have been studies that apply dimensionality reduction within their works. In most cases, objectives that are considered to be redundant to the optimization process are removed from the objective set permanently. In this research, a nonconflicting objective function isn't removed for good. The DRMOO algorithm also leaves room for error in case an objective is wrongly labelled as redundant. Here, the objective function is removed from its current objective subset and placed within another objective subset. The algorithm continues to test if the objective function is redundant within its new set. This way, there is a higher chance that the objectives within an objective subset are nonredundant and conflicting in nature. Each objective is also given equal importance

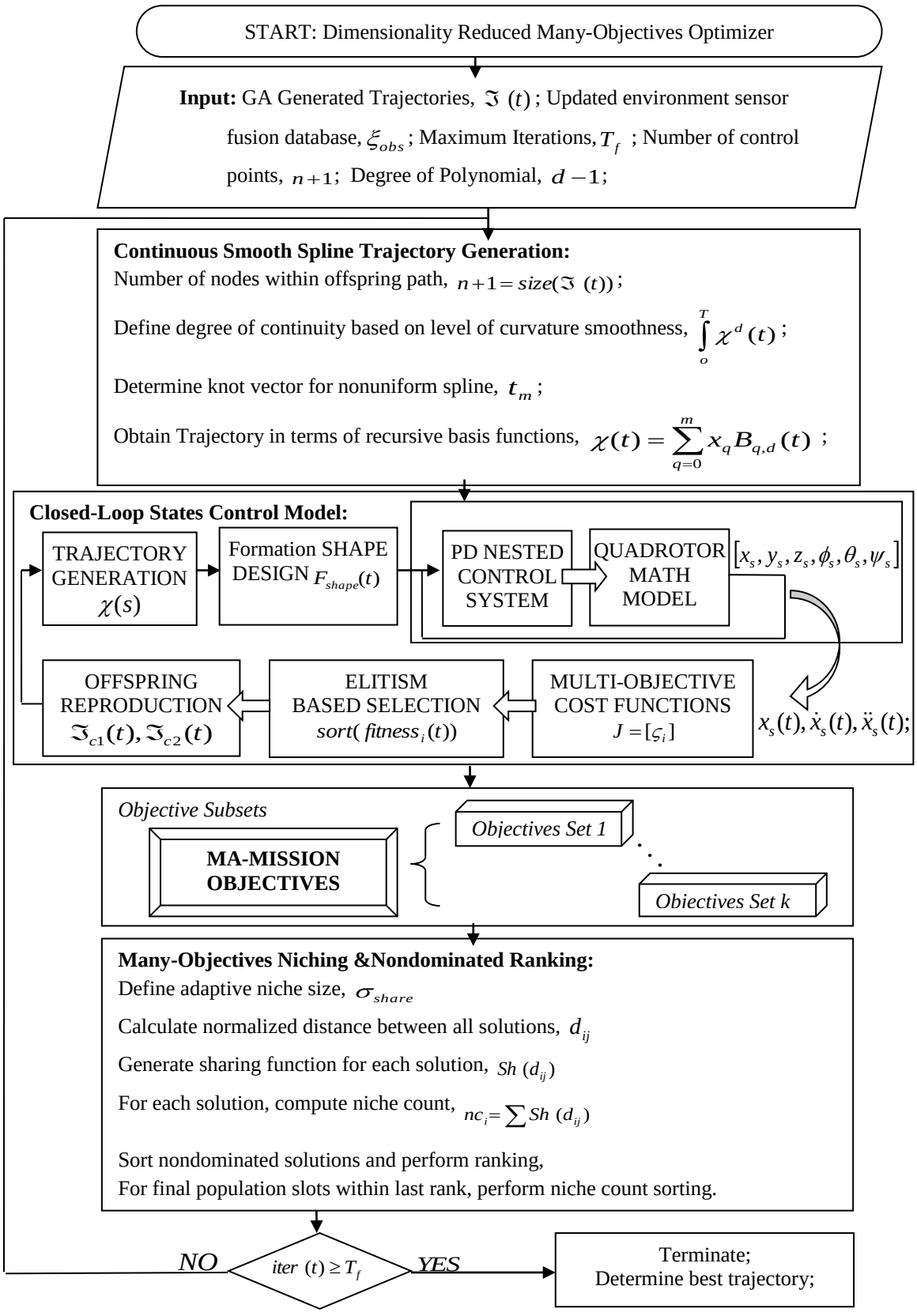


Fig. 4.21 Many-Objectives Optimization Algorithm Flow Chart.

through Partial Pareto optimization. Each of the many objectives is still equally important despite being broken into smaller subsets. This is executed by rotating the objective subset that is being used within the algorithm. Some studies that have applied this concept randomly create objective subsets. In this case, the algorithm is less efficient because there is a high chance of nonconflicting objectives being within the same subset. In this research, a more strategic manner of creating objective subsets is utilized through dimensionality reduction.

The algorithm begins with the full set of objectives. At set interval points, an objective function that is considered to be nonconflicting is removed from its original set and placed into a new subset. This process continues until there are a few objective subsets. As the algorithm progresses, these objective subsets are used in rotation. At the end of each interval, the full objective set is reintroduced. The application of both the full objective set and subsets allow the algorithm to perform both local and global optimization simultaneously. Also, no objectives are fully eliminated. This is advantageous in cases where an error has been made in determining nonconflicting objectives.

First, the full objective set, J_{full} is defined as the initial set prior to partitioning. Next, the partitioning of the objective functions is initialized. An objective function is selected at random from the current objective set, $\varsigma_n(t)$ and removed to create, J_{new} . The ratio, κ is used to identify if the chosen objective function doesn't conflict with the other objectives within its set [133]. This ratio is applied based on the concept that if an objective is nonconflicting, it doesn't contribute to the ranking process. The ranking of the solutions is possible without the inclusion of a nonconflicting objective within its set. Thus, the number of dominant solutions remains within the same range despite the removal of a nonconflicting objective.

$$\kappa = \sum N_{J\ removed} \Big/ \sum N_{J\ current}, \kappa \in [0,1] \tag{4.47}$$

where $N_{J_{current}}$ = total dominant solutions within current set

$N_{J_{removed}}$ = total dominant solutions if the objective function is removed from its current set

Parameter $\kappa_{threshold}$ determines the cut-off point where the removal of an objective function is acceptable. Upon exceeding the threshold value, the objective function perceived to be redundant is removed from its current set and merged into a new set.

$$J_{new}(t) = [\varsigma_n(t)] \quad if\ \kappa \geq \kappa_{threshold} \tag{4.48}$$

As the algorithm progresses, more objective functions may be found redundant within the new. Thus, the initialization of more new sets, J_{sets} begin to hold more combinations of objectives. This is only performed after every constant number of iterations as to partition the objectives at a steady rate. As the DRPPD progresses, each objective set is always inspected for nonconflicting objectives.

$$J_{sets}(t_f) = [J_{full}(t)\ ;\ J_{subset\,1}(t);\ J_{subset\,2}(t);_{....}\ J_{subset\,p}(t)] \tag{4.49}$$

where p = number of objective subsets

The partitioning of the cost functions is halted when each set is reduced to a minimum of three objectives. Both the full and multiple subsets of the objectives are used in rotation as the algorithm progresses through iterations. The population set at each generation is the combination of the previous globally and current locally optimized solutions.

The approximation of optimal solutions is performed through the classifications of solutions. The decision vector, $t \in \Omega$ is dominant for a many-objective optimization problem if there exists for no other, $t* \in \Omega$

$$\varsigma_n(t) \le \varsigma_n(t*),\ n \in \{\ 1,\ ...,\ N_f\ \} \tag{4.50}$$

where Ω = full solution set

The solutions are ranked and stored into the new generation's parent population. In some cases, the dominant solutions do not fill the parent population. Thus, niching is performed on the remaining population in order to determine which are most diverse.

4.5.3 ADAPTIVE NICHING

In this study, the removal of solutions that are similar and maintenance of dissimilar solutions is executed through adaptive niching. Based on experimentation, the application of adaptive niching parameter by varying the radius of a niche, σ_{share} through each generation produces a better balance between trajectory diversity and dominance in comparison to a constant niche radius [139]. At each generation, the distance between each solution and its nearest neighbour is determined. The average of nearest neighbour distances, d_{avg} of all solutions is,

$$d_{avg} = \min_{j \ne i}(\left\|\varsigma_i - \varsigma_j\right\|)\ ,\ (\ i, j = 1,2,\ldots N_{pop}\ ) \tag{4.51}$$

The average distance of the solution population is then used to determine the current niche radius.

$$\sigma_{share} = \begin{cases} c & if\ N_{pop} < 2 \\ \sum_{i=1}^{N_f} d_{avg} \Big/ N_{pop} & others \end{cases} \tag{4.52}$$

The sharing function, $Sh(d_{ij})$ creates the comparison between two solutions as to sharing of each optimum. The summation of the sharing function defines the niche count, nc_i for each solution is then used as a measure of the percentage of the total solutions that belong to a certain optimum. The adaptive adjustment of the niche radius is based on the changing sizes of clusters within the objective space. It encourages the degradation of crowded solutions and the enhancement of cluster representative solutions. Thus, the adaptive niche radius allows the maintenance of clusters of solutions that represent different optima.

The next chapter will show the effectiveness of the hybridized trajectory planner in terms of producing a collection of optimized and diverse set of solutions. The MA-RRF, GA and DRMOO will be applied collectively for both the MA-Spread and MA-Formation missions.

4.6 SUMMARY

This chapter defines the 16 objective functions that are applied for both the MA-Spread and MA-Formation application. There are 8 costs that are applied for both missions. There are also 8 more costs that are designed specifically for each mission.

The next section described the optimization process within the MA-Spread mission. The algorithm is designed to optimize the paths for individual agents on separate flight directions. Here, no agent is given priority over the other. Thus, the performance of any agent isn't degraded for the sake of the other. This is achieved by optimizing the combination of multi-agent paths at each generation. The MA-Formation formation configuration design was also defined within this chapter. A dynamic formation planner is run before the optimization process. High resolution obstacle detection and free space contour definition is paramount to the efficient planning of formation shapes. Thus, the dynamic formation planner designs formation shapes that allow the agents to morph into the free space contour seamlessly. The algorithm aims to derive well minimized reference trajectories that encourage formation maintenance and minimize complexity. Here, emphasis is placed on collision avoidance, formation rise time and the number of

variations of formation design per path. Lastly, the DRMOO ranking algorithm is defined in detail. This algorithm combines both partial dimensionality sorting with full high dimensionality optimization. The algorithm is designed to focus on the minimization of all objectives as opposed to mapping the Pareto front with full accuracy.

In the next chapter, we then utilize the many-objectives optimization algorithm to show its versatility and robustness within the MA-Spread and MA-Formation application. The minimizations of the objective functions within both missions are analysed. The results will show the algorithm's ability to sort and rank the path combinations at each generation. The final generation's well minimized and diverse multi-agent trajectories within the three test environments are also presented.

CHAPTER 5: MULTI-AGENT QUADROTOR UAVS IN SPREAD AND FORMATION FLIGHT MISSIONS

The previous chapters have presented the different subsections that make up the offline long-range multi-agent quadrotor UAV path planning algorithm. Firstly, Chapter 3 showed how the combining MA-RRF and GA can lead to a path planner that uses a multi-agent UAV system to its advantage. These algorithms can generate a large collection of hybridized paths for multiple agents across many generations. Next, Chapter 4 presented 16 different objective functions that will be used by the DRMOO algorithm to rank the multi-agent trajectories. The feasibility of each designed path and the values for each objective function can be estimated by the parallel run multi-agent quadrotor UAV control system. Lastly, this chapter will evaluate the effectiveness of the path planning and optimization algorithm within the MA-Spread and MA-Formation missions. Both applications are fully tested across three environments which are the high-rise cityscape, highly cluttered indoor and mountainous terrain.

The first section presents the simulation results for the MA-Spread spatially spread flight scenario. In MA-Spread, the algorithm faces the challenge of navigating 4 agents whilst gathering sensory data through full exploration across different terrains. Here, the various combinations of paths produced by the algorithm at the final iteration for all three environments are shown. The second part of this chapter presents the simulation results of the trajectory planner within the MA-Formation mission. This application requires the algorithm to generate a collection of well minimized formation reference trajectories for 8 quadrotors. In MA-Formation, the MA-RRF paths of one agent are applied as the initial formation reference trajectory population. These reference trajectories will be used as the input for the dynamic formation planner. The results within this chapter will show if the dynamic formation planner can create fast and adaptive formation shapes across all environments. It also determines if the formation planner can generate independent trajectories for each agent within its constantly changing formation structure.

This chapter showcases a variety of results and findings within both the MA-Spread and MA-Formation missions. In the beginning, the results of the optimization algorithms' operators are analysed for both applications. Here, the development of the dimensionality reduced ranking process and adaptive niching are shown. It is important to note that this work isn't attempting to obtain a solution set that converges towards the Pareto frontier. It is more focused on applying the concepts of many optimization algorithms. This project implements the Pareto optimal ranking process to maintain good solutions within a population. Thus, the goal of the algorithm is to continue to minimize the objective values whilst maintaining a diverse solution population. The multi-agent quadrotor UAV trajectories within each mission must have different strengths and weaknesses so that the

end user is presented with a diverse collection of solutions. Next, the values for all 16 objectives across all generations are presented. The final generation is chosen based on the minimization of all objectives as well as the diversity of the trajectories. The final values for each objective function is shown and analysed. Finally, the path combinations that are maintained within this final generation are shown through three-dimensional imagery. Lastly, the objective values for each combination is presented in table form.

This chapter aims to show the variety of information that will be presented to the end user at the end of the trajectory optimization process. It defines the importance of providing easy to understand knowledge for the end user so that they can make a knowledgeable choice for their mission. It also allows the end user to compare each path option visually. It also allows the end user to take into consideration their post-processing objective preferences. The data that is presented here is only the results of the final generation out of many iterations. It is important to note that there are thousands more options in the other generations if the end user requires even more variety.

5.1 MA-SPREAD DIMENSIONALITY REDUCED MANY-OBJECTIVES PARAMETERS

Firstly, the progression of the adaptive operators is shown in Figure 5.1-5.2 for Cityscape 1, Indoor 1 and Mountain 1. The constant that were used in the MA-Spread application are shown in Table 5.1. Each objective subset is run for 5 iterations with a singular run of the full set of objectives. These figures show the changes that occur in terms of dominance and diversity.

Figure 5.1 shows that there are reoccurring peaks where a large percentage of the population are nondominated solutions. Here, most solutions are ranked as dominant when the full objective set is considered. Thus, the only manner of comparing solutions is through its level of diversity. This causes the members of each generation to be filled with suboptimal solutions. Convergence towards optimality is possible with the introduction of objective subsets during intervals. This can be seen through the existence of lower peaks that occur during the rotation of objective subsets. The ranking of solutions based on dominance is possible given that a smaller number of solutions are nondominated. As long as these low peaks frequently dip below half of the population size, the future generation parent population remains partially filled with nondominated solutions.

Adaptive niching is applied for secondary ranking of remaining solutions. Figure 5.2 and shows the changes of niche radius size by adapting to the diversity of the current population. If a constant value for niche radius was used, the increase or decrease in distances between solutions as the generations' progress won't be represented. With many-

objectives optimization, the sizes of clusters will vary as the algorithm evolves. With adaptive niching, solutions that are similar in estimated objective values are gradually removed from the population leaving representative solutions of various clusters to survive to the next generation. Even though the distances between clusters do not increase immensely, maintenance of the niche radius within a small range shows the preservation of diverse solutions across generations.

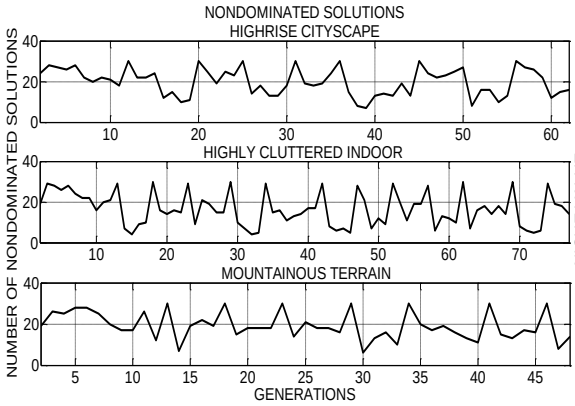


Fig. 5.1. Number of dominant solutions within the population.

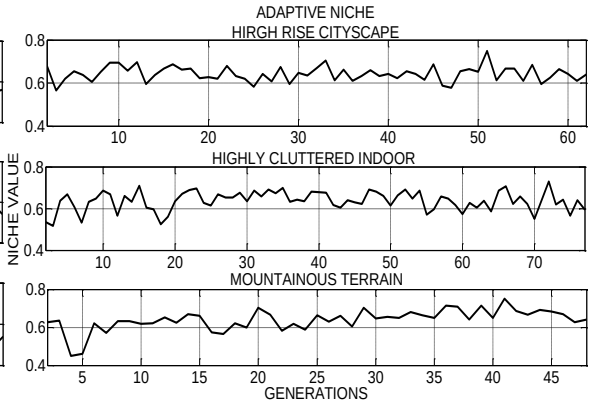


Fig. 5.2. Adaptive niche radius across generations.

5.2. MA-SPREAD ACROSS A HIGH-RISE CITYSCAPE

The cityscape environment holds many unique challenges for trajectory planning and optimization algorithm. This test space has a large amount of narrow passages that are similar to a maze. The buildings are tall and don't allow the agents to fly above it. Thus, the designed paths must not collide with these buildings. It must also contain minimal aggressive turns across the building's sharp corner bends.

There are three varieties of information that are provided to the end user for Cityscape 1. These are tables, graphs and imagery. Figure 5.3-5.4 shows the progression of the objective values at each generation. Next, Table 5.2 shows the average cost values of the entire trajectory population for the final generation. The paths for all four quadrotors are presented in Figure 5.5. Lastly, Figure 5.6-5.8 shows the various numbers of combinations within the final population. Table 5.3 is attached to compare of these combinations in terms of their trade off values for all objective functions. Post determination of the best combination for the end user needs can be performed based on these values. Here, we evaluate the best and worst choices between the presented combinations. This section shows that the hybridized algorithm successfully produces a collection of path combinations. These combinations are well optimized and there are different advantages to each choice.

5.2.1 MANY OBJECTIVES VALUES

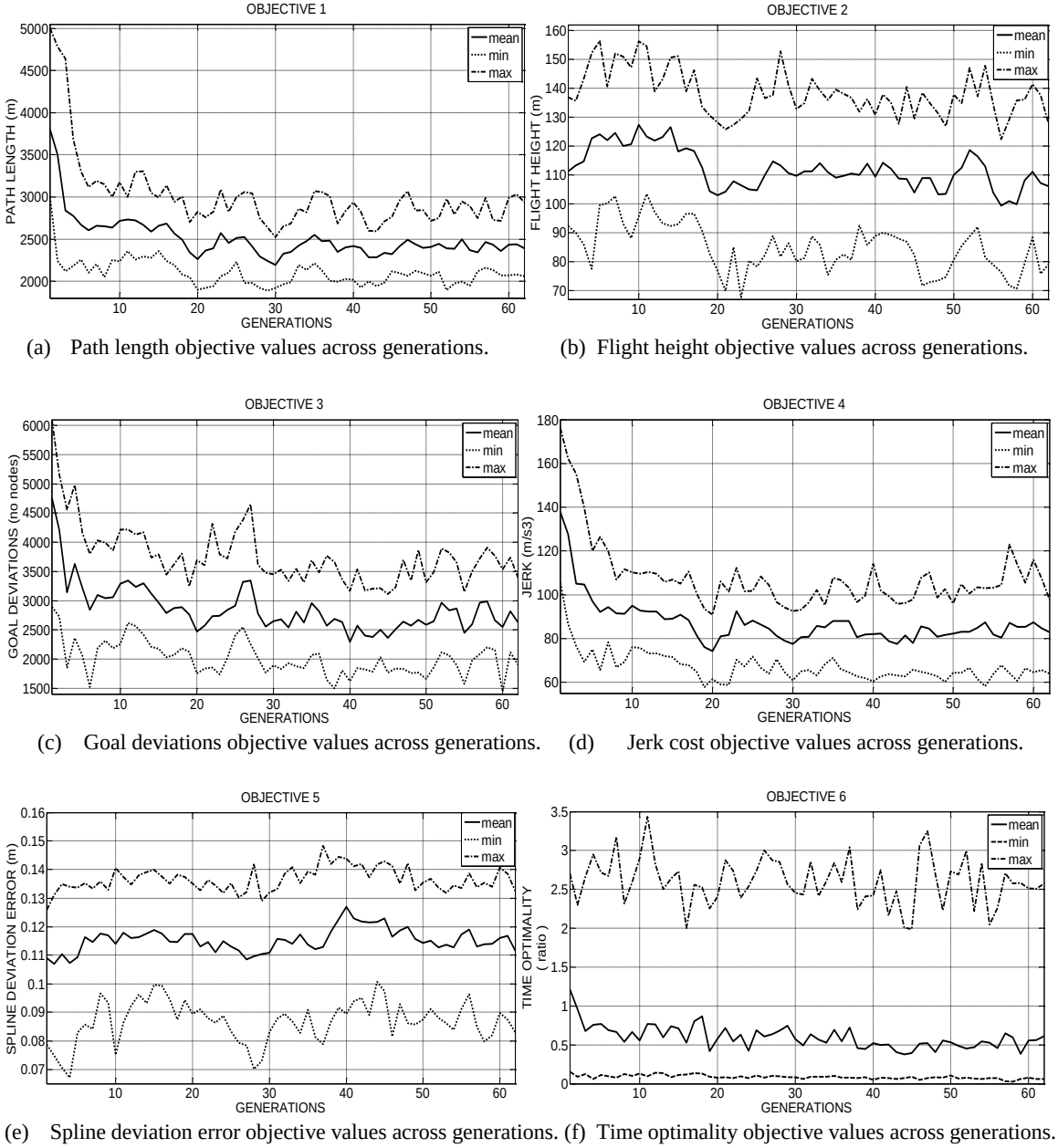
As previously defined, each environment is run for 100 generations. It is important that all the objectives are given the time to optimize their values. The final generation is chosen based on the minimization of the many objectives. The final generation can be chosen if the cost values do not decrease any further. The values for each objective are easily obtained from the shared database. The choice for the final generation is also dependent on the level of diversity within the path population. The end user can see if the paths within the population are varied in direction through the imagery that is provided.

Figure 5.3-5.4 shows the progression of all objectives across the generations for the Cityscape 1 environment. The graphs show that initially most objectives decrease at a fast rate. The values of the cost functions then decrease at a slower rate as the generations' progress. The final generation is set at the 62^{nd} iteration. This is because objective 2,6,8,9 and 12 only meet their minimal point after the 50^{th} generation. There is no increase in the mean values of all objective functions. The graphs also show that the values for objective 2 and 5 are well maintained. The Cityscape 1 environment has extremely narrow passages. The agents often fly across similar areas. Thus, in most combinations the path nodes are well spread across the height of the environment. Due to this, it can be tough to minimize the height cost. The value of the spline deviation error isn't an issue as it is still maintained at a very small value. This cost is also difficult to optimize because it is dependent on the control system as well as the node-to-node distance within a path. On the other hand, all the other costs are well minimized.

Table 5.2 shows that the mean values of all the 12 objectives at the 62nd iteration. The data shows that the cost values are well minimized or maintained within the final generation. It shows that the spline deviation cost is maintained across generations. The altitude cost is slightly minimised by 5%. Similarly, the number of safety zone breaches has a minor reduction of 2%. The amount of highly explored space is reduced by 12% whereas the number of sensory data overlap is minimized by 22%. These values aren't as high as the reduction within the other environments because of its constricted nature. There isn't much space for the agents to move between the buildings. Despite this, Figure 5.5 shows that the narrow passages are well explored.

All the other objectives are well minimized with a more than 30% reduction in value. The path length for the trajectory combinations is reduced by 37%. This shows that the paths are much shorter than the initial MA-RRF paths. The number of goal node deviations is also minimized by 45%. This proves that the paths move more directly to the goal node as well. Next, the jerk cost for the path combinations is lessened by 40%. The lower value of the jerk cost shows that the paths hold less aggressive turns across the

building's sharp corners. The cost for time optimality shows that the agents are able to fly at a speed that is 50% faster. Similarly, the flight time has reduced from about 50 to 30 minutes per agent. The path combinations allow the agents to maintain connectivity at 51% more than the initial population. Lastly, there is 77% less possible collisions between the quadrotors in a team. Table 5.3 shows that nearly all combinations have zero possible collisions except for a few. The data shows that the trajectory optimization algorithm for the MA-Spread mission across the Cityscape 1 environment successfully minimizes the cost functions.

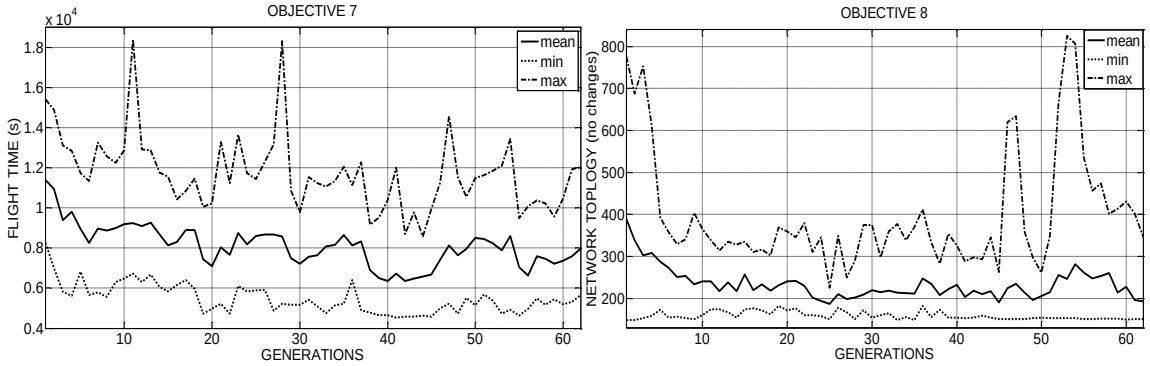


(a) Path length objective values across generations. (b) Flight height objective values across generations.

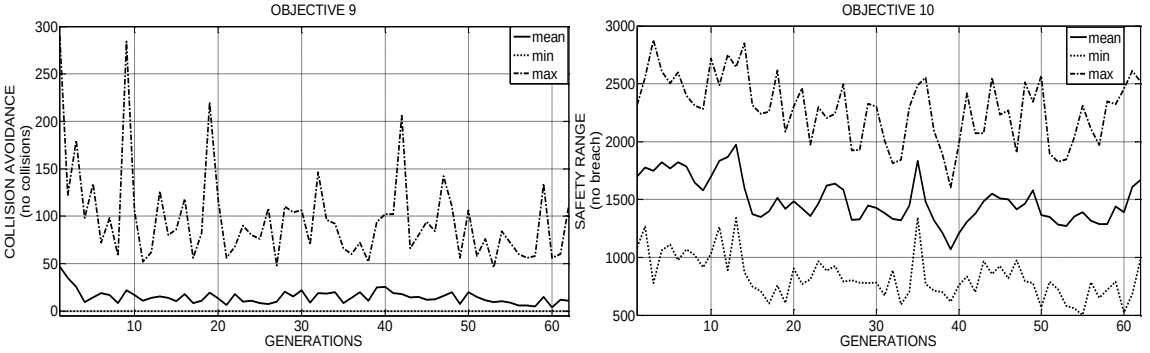
(c) Goal deviations objective values across generations. (d) Jerk cost objective values across generations.

(e) Spline deviation error objective values across generations. (f) Time optimality objective values across generations.

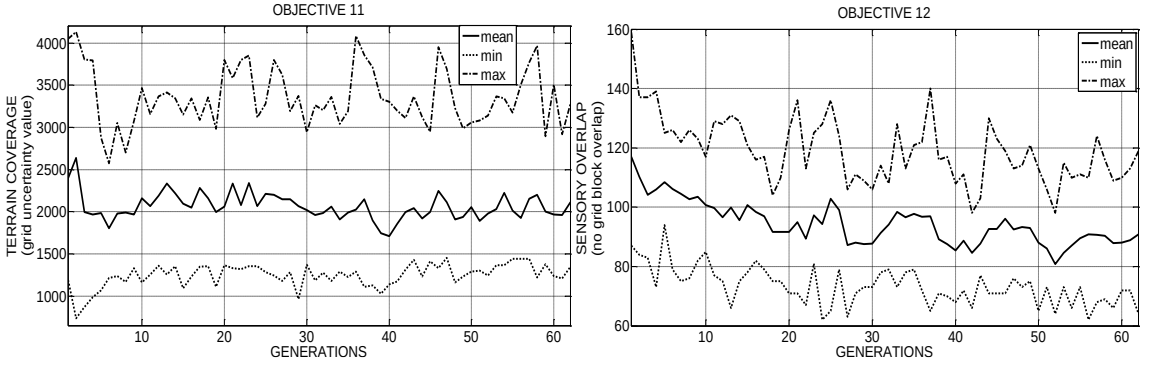
Fig. 5.3. MA-Spread Cityscape: Progression of the objectives 1-6 across generations.



(a) Flight time objective values across generations. (b) Connection breaches objective values across generations.



(c)Possible collisions objective values across generations. (d) Safety zone breaches objective values across generations.



(e) Uncertain grid coverage objective values across generations. (f) Sensory overlap objective values across generations.

Fig. 5.4. MA-Spread Cityscape 1: Progression of the objectives 7-12 across generations.

TABLE 5.1: MA-SPREAD PARAMETERS

Description	Value	Description	Value	Description	Value	Description	Value
Number of Paths Combinations	30	Minimum Agent-to-Agent Distance	$10m$	Population size	30	$r_{external}$	$50m$
Number of agents	4	Number of Gaui 330X-S Agents	2	Selection Rate	0.5	[$d_{ij,min}$, $d_{ij,max}$]	[-5, 5] m
Terrain Grid Block Size	$30m^3$	Gaui 330X-S Agents Maximum Fuel	15mins	d_{goal}	$50m$	$t_{collision}$	1
Network Decay Range	$0.75\,\xi$	Number of Fyetech Agents	2	d_{link}	$10m$	$\kappa_{threshold}$	0.8
Obstacles Boundary Size	$\xi_{obs} \pm 6m$	Fyetech Agents Maximum Fuel	10mins	$d_{similar}$	$10m$	c	1

TABLE 5.2: MA-Spread HIGH RISE CITYSCAPE 1 MANY-OBJECTIVES OPTIMIZATION RESULTS.

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION *(m)*	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE *(no free) space grids*	SENSORY OVERLAP (*no grid block) overlap*
1	max	5006.70	136.90	6136.00	176.24	0.13	2.70	15420.00	776.00	290.00	2314.00	4050.90	159.00
	mean	3802.20	111.35	4761.50	137.77	0.11	1.21	11400.00	388.80	47.00	1698.30	2398.10	116.97
	min	2998.10	92.35	2895.00	105.45	0.08	0.16	8259.40	148.00	0.00	1087.00	1188.00	87.00
62	max	2937.30	127.93	3389.00	97.61	0.13	2.58	12036.00	346.00	112.00	2517.00	3281.70	119.00
	mean	2393.10	106.21	2626.30	82.81	0.11	0.61	7986.50	191.90	10.87	1668.60	2111.40	90.87
	min	2058.80	79.16	1896.00	63.92	0.08	0.06	5660.10	150.00	0.00	989.00	1344.90	64.00
	%	37.06	4.62	44.84	39.89	0.00	49.59	29.94	50.64	76.87	1.75	11.96	22.31

5.2.2 TRAJECTORY POPULATION

The MA-Spread mission requires the optimization algorithm to minimize the cost of the entire team of agents as a singular entity. Each path combination is sorted and ranked according to their collective objective values. Thus, the number of unique paths per agent at each generation is dependent on the path combinations that survive the selection process.

Figure 5.5 shows the unique paths that have been maintained within the 62nd population. The individual paths for each agent are presented in a different colour. There are no obstacle collisions despite the large number of buildings. Similarly, the paths are capable of smoothly bending around sharp corners. The usage of minimal jerk fifth order splines is proven to be useful in minimizing the number of aggressive manoeuvres. The images show that the paths for an agent are diverse and well spread across the terrain. This means that the level of diversity that was seen within the initial MA-RRF paths has been maintained across generations. It also shows that each path combination is diverse as well. Each agent's paths could create thousands of possible combinations. The optimization algorithm must be capable of only maintaining combinations that produce minimal cost values. Figure 5.6-5.8 shows how these unique paths create different path combinations. The 30 path combinations make up the entire population for the final generation.

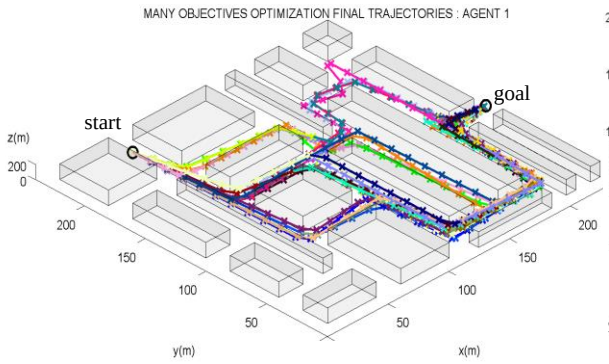
The paths within the Cityscape 1 environment for the four agents are shown through various colours. Figure 5.6-5.8 shows the path nodes of the first to last agent which is shown in magenta, red, blue and green. In many cases, the best choice in terms of an objective isn't apparent to the naked eye. Firstly, this is because the imagery needs to be in three dimensions to view the progression of the path in all directions. Secondly, this is due to the adaptive nature of the path nodes. The objective values will differ for each combination since the paths are padded based on their node to node distances as well as curvatures. It is the data that is provided to the end user that clearly shows the trade-off values for each choice.

The first combination has many nodes across the lower part of the terrain. It has the minimum altitude cost. On the other hand, combination 29 has the most nodes across higher parts of the Cityscape 1 environment. Thus, it has the maximum altitude cost. Combination 5 produces flight paths that allow the agents to fly faster than quadrotor's average velocity. This option has the minimal value in terms of time optimality and as well as the best coverage of uncertain terrain. Combination 6 is comprised of paths that produce the shortest paths, least goal deviations, lowest jerk cost and minimum fuel consumption. Figure 5.6(f) shows that all the agents fly directly towards their destinations except for agent 2. Table 5.3 shows that most of the combinations produce trajectories that have zero possible collisions. Choosing combination 10 leads to paths that have the highest amount

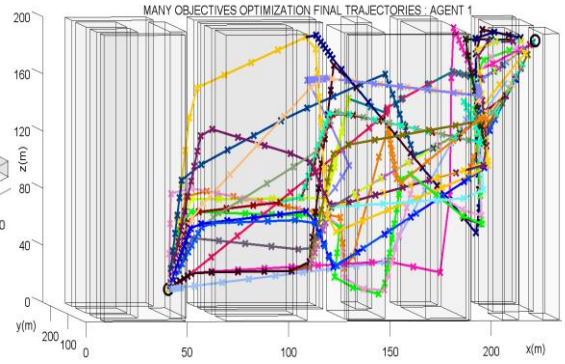
of possible agent-to-agent collisions. Figure 5.6(j) shows that there are many path subsections where the agents are flying within proximity of each other.

The option with the least number of communication network decay is combination 12. Here, the agents do not fly further than the communication range at each point in time. Option 16 has the least number of sensory data overlap whereas option 20 has the most. It can be seen in Figure 5.7(f), the paths progress across different sections of the terrain without too many overlaps. Similarly, Figure 5.7(j) shows that many nodes cross the same areas within the terrain. Combination 24 has paths with many consecutive sharp turns which contribute to a high jerk cost. Lastly, option 30 produces paths with minimal possible collisions and safety zone breaches. Figure 5.8(j) shows that many nodes are places away from the boundaries of the buildings. Each path combination is well minimized and has its own advantages. Thus, the end user can make a final decision based on their priorities.

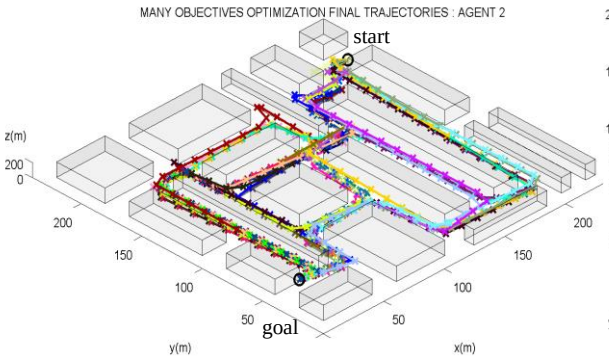
As expected, all combinations have been well minimized as seen with the reduction or maintenance of mean values in Table 5.2. This proves that the many-objectives algorithm successfully optimized the trajectory planning process for spread flight within the Cityscape 1 environment through the approximation of the Pareto front. The hybridized algorithm may not have fully extracted Pareto optimal solutions but has provided a diverse as well as minimized collection of trajectories. Based on post processing preferences, the strengths of each combination will aid in the final solution choice.



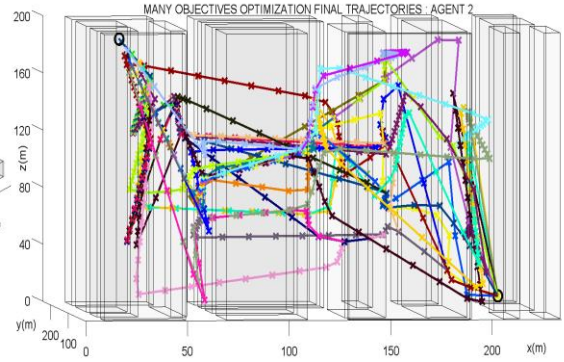
(a) Top view of agent 1's paths within the final generation.



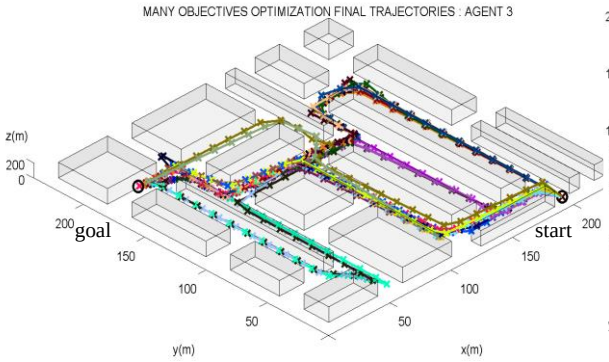
(b) Side view of agent 1's paths within the final generation.



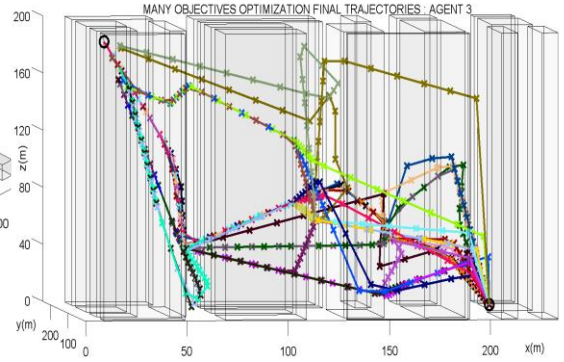
(c) Top view of agent 2's paths within the final generation.



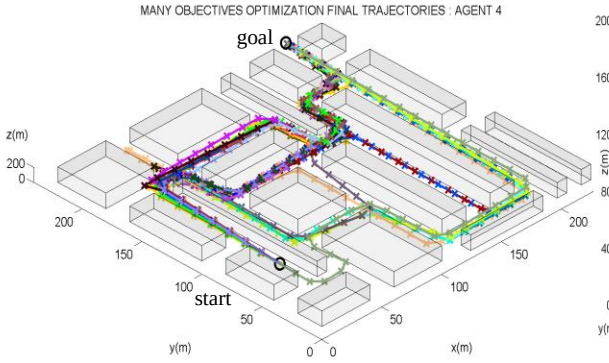
(d) Side view of agent 2's paths within the final generation.



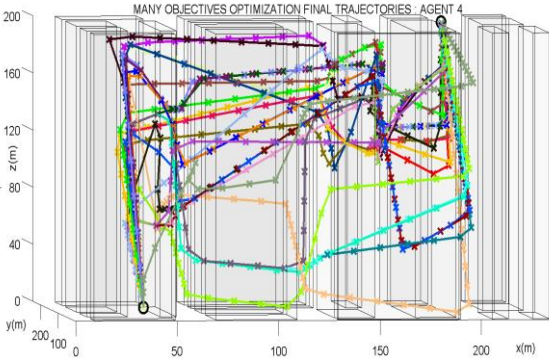
(e) Top view of agent 3's paths within the final generation.



(f) Side view of agent 3's paths within the final generation.

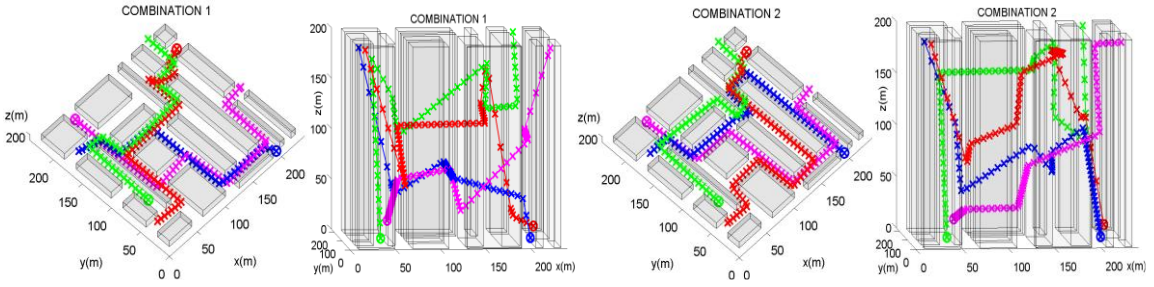


(g) Top view of agent 4's paths within the final generation.



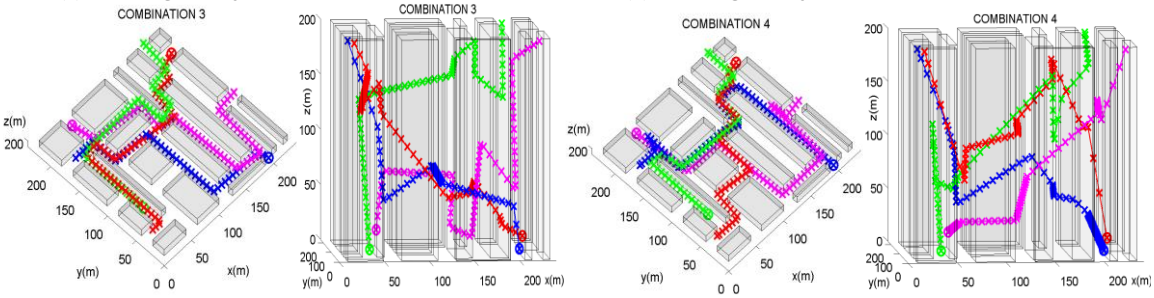
(h) Side view of agent 4's paths within the final generation.

Fig. 5.5. MA-Spread Cityscape 1: Final generation's multi-agent unique trajectories.



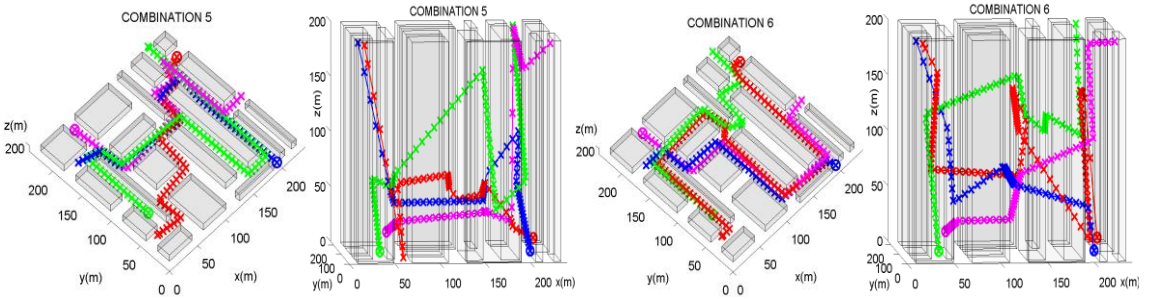
(a) Multi-agent trajectories within combination 1.

(b) Multi-agent trajectories within combination 2.



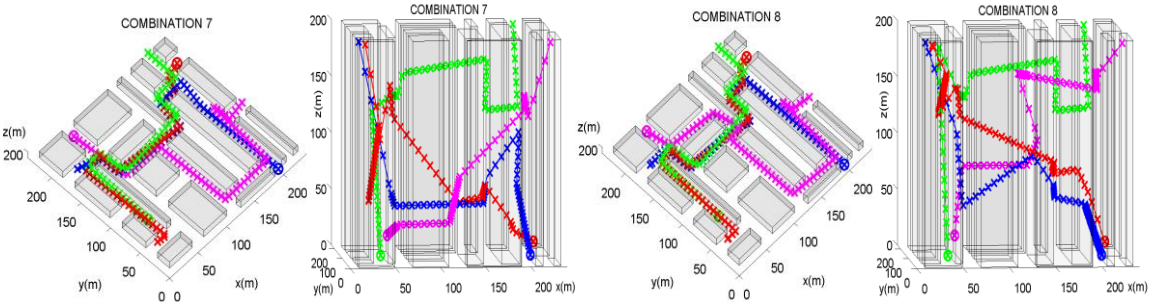
(c) Multi-agent trajectories within combination 3.

(d) Multi-agent trajectories within combination 4.



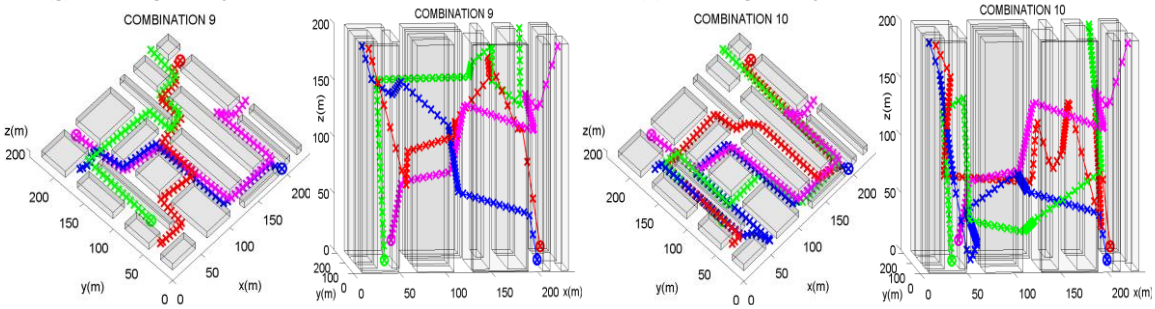
(e) Multi-agent trajectories within combination 5.

(f) Multi-agent trajectories within combination 6.



(g) Multi-agent trajectories within combination 7.

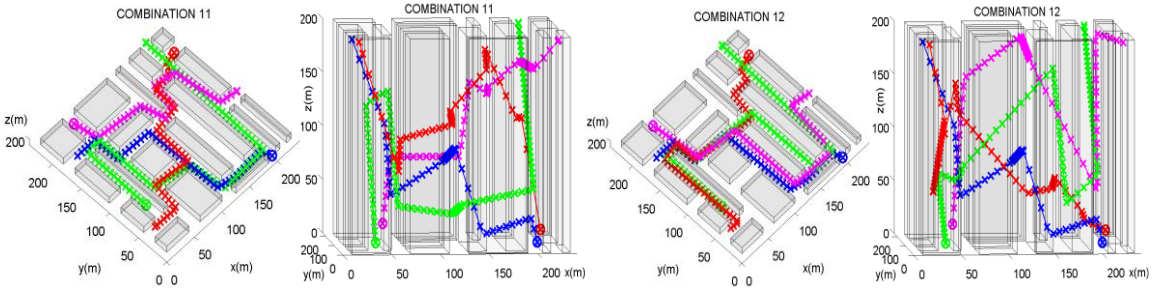
(h) Multi-agent trajectories within combination 8.



(i) Multi-agent trajectories within combination 9.

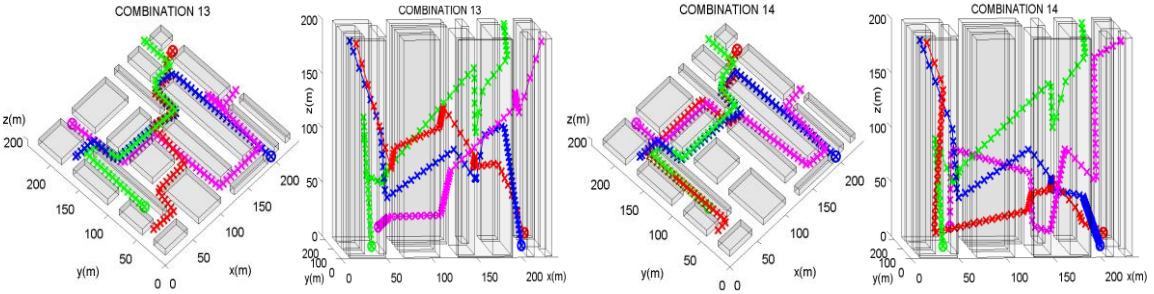
(j) Multi-agent trajectories within combination 10.

Fig. 5.6. MA-Spread Cityscape 1: Multi-agent path combination 1-10 for the final generation.



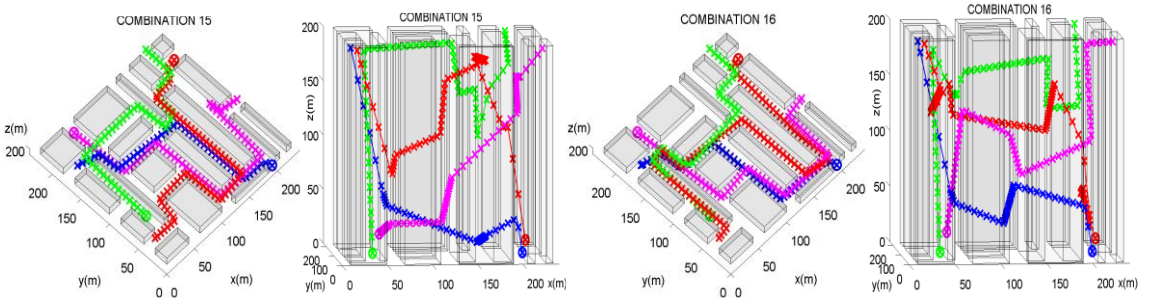
(a) Multi-agent trajectories within combination 11.

(b) Multi-agent trajectories within combination 12.



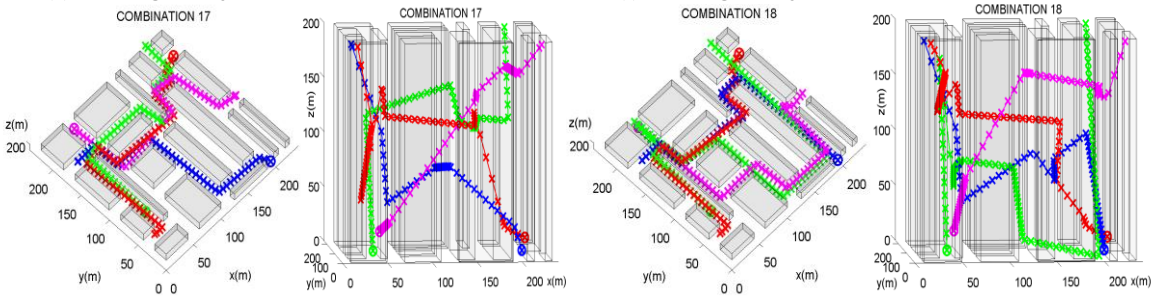
(c) Multi-agent trajectories within combination 13.

(d) Multi-agent trajectories within combination 14.



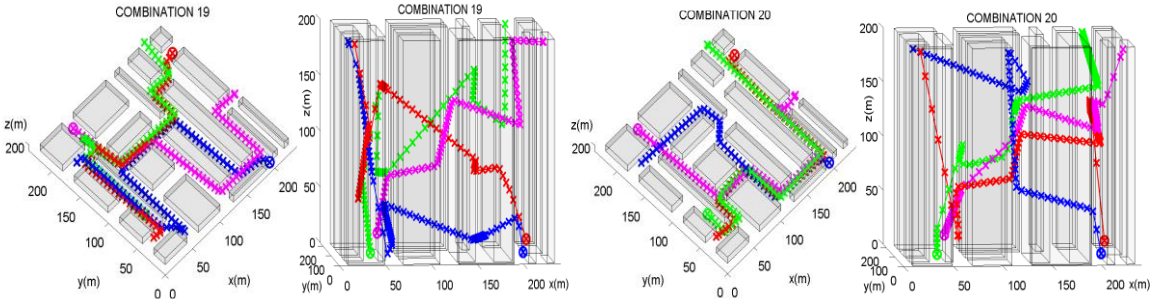
(e) Multi-agent trajectories within combination 15.

(f) Multi-agent trajectories within combination 16.



(g) Multi-agent trajectories within combination 17.

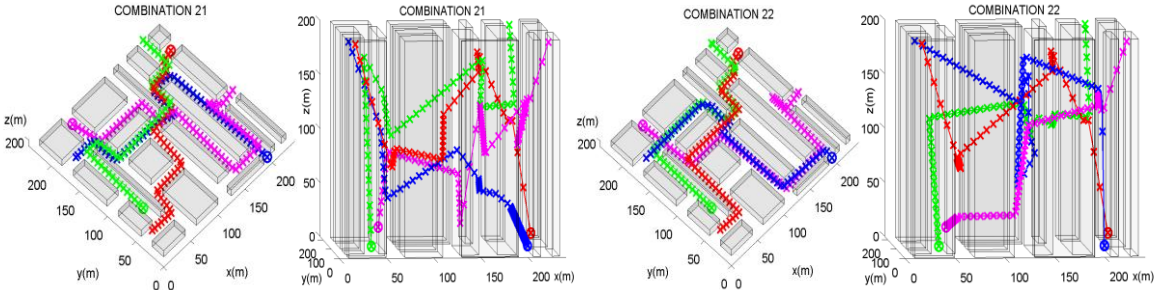
(h) Multi-agent trajectories within combination 18.



(i) Multi-agent trajectories within combination 19.

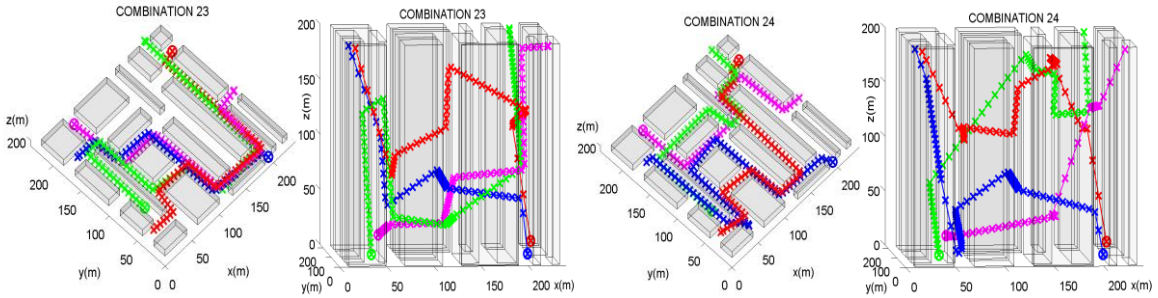
(j) Multi-agent trajectories within combination 20.

Fig. 5.7. MA-Spread Cityscape 1: Multi-agent path combination 11-20 for the final generation.



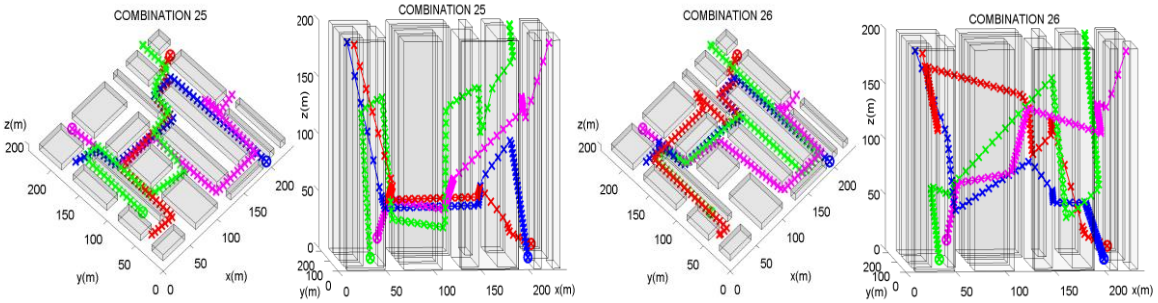
(a) Multi-agent trajectories within combination 21.

(b) Multi-agent trajectories within combination 22.



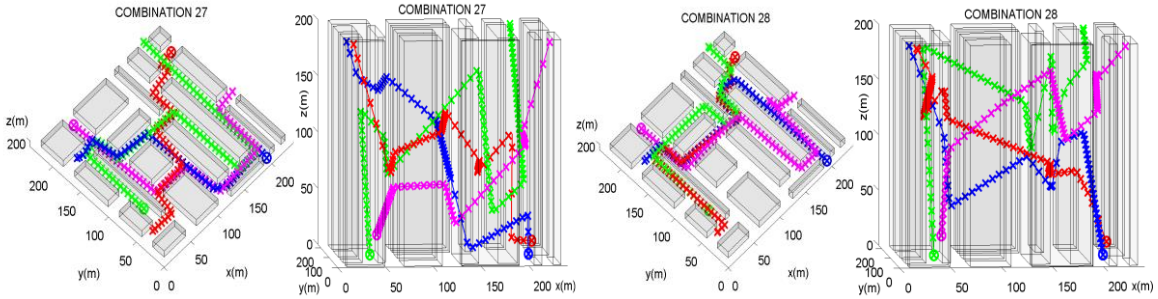
(c) Multi-agent trajectories within combination 23.

(d) Multi-agent trajectories within combination 24.



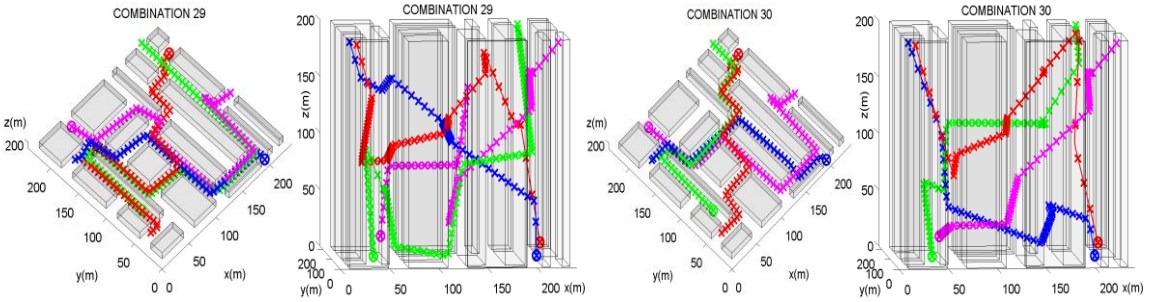
(e) Multi-agent trajectories within combination 25.

(f) Multi-agent trajectories within combination 26.



(g) Multi-agent trajectories within combination 27.

(h) Multi-agent trajectories within combination 28.



(i) Multi-agent trajectories within combination 29.

(j) Multi-agent trajectories within combination 30.

Fig. 5.8. MA-Spread Cityscape 1: Multi-agent path combination 21-30 for the final generation.

TABLE 5.3: MA-Spread HIGH RISE CITYSCAPE 1 FINAL GENERATION'S RESULTS.

CITYSCAPE 1 Spread COMBINATION	PATH LENGTH (*meters*)	ALTITUDE COST (*meters*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE ERROR (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE (*% free space*)	SENSORY DATA OVERLAP (*grid block overlap*)
1	2386.49	79.16	2287.00	76.92	0.13	0.28	6759.67	153.00	0.00	1636.00	1578.29	72.00
2	2937.28	94.22	2733.00	90.12	0.12	0.13	9747.77	201.00	0.00	1401.00	1446.83	99.00
3	2227.84	104.82	2401.00	67.33	0.13	0.12	7034.58	160.00	0.00	1315.00	1777.68	98.00
4	2438.70	94.48	2487.00	72.41	0.13	0.23	5988.06	155.00	0.00	1245.00	1500.64	98.00
5	2653.05	112.69	2685.00	85.40	0.12	0.06	7422.19	269.00	8.00	1381.00	1344.90	107.00
6	2058.81	91.50	1896.00	63.92	0.12	0.19	5660.12	158.00	0.00	1607.00	1699.26	86.00
7	2524.89	105.50	2119.00	84.40	0.12	0.11	6503.21	153.00	0.00	1352.00	2022.23	88.00
8	2554.88	88.43	2888.00	84.03	0.12	0.36	9734.61	160.00	0.00	2193.00	1905.15	96.00
9	2406.43	96.00	2383.00	84.69	0.11	0.42	7669.07	238.00	0.00	1459.00	2194.56	95.00
10	2157.09	113.96	1999.00	79.17	0.11	0.37	6348.65	172.00	112.00	1899.00	2015.74	78.00
11	2318.55	104.72	2818.00	75.09	0.13	0.15	6005.96	226.00	0.00	2096.00	1603.62	102.00
12	2656.35	81.70	2656.00	85.65	0.13	0.14	7215.72	150.00	0.00	2062.00	1697.65	104.00
13	2121.26	104.57	2429.00	74.47	0.11	0.16	6734.19	171.00	0.00	2078.00	2072.60	99.00
14	2629.37	104.10	2757.00	92.07	0.10	1.91	9524.38	213.00	0.00	1066.00	2337.02	76.00
15	2415.26	119.05	2565.00	86.05	0.11	0.13	6323.50	302.00	0.00	1266.00	1747.79	85.00
16	2095.86	93.32	2114.00	69.75	0.12	0.28	5795.51	346.00	0.00	1268.00	1437.06	64.00
17	2510.99	98.32	2581.00	83.94	0.12	0.31	7854.32	158.00	0.00	1070.00	2171.54	93.00
18	2170.38	122.61	2568.00	80.13	0.11	2.31	8489.86	212.00	0.00	1737.00	2328.13	89.00
19	2747.60	106.84	2813.00	83.62	0.11	0.48	12035.59	195.00	0.00	2517.00	1524.13	118.00
20	2604.30	99.61	3008.00	95.05	0.12	0.63	7332.55	307.00	0.00	1227.00	1880.87	119.00
21	2147.71	122.60	3068.00	90.74	0.08	2.58	8964.75	180.00	60.00	1528.00	3143.11	110.00
22	2162.45	111.69	2186.00	72.45	0.11	0.31	7369.78	160.00	38.00	1727.00	2325.91	87.00
23	2425.23	106.44	2492.00	90.72	0.11	0.15	7881.72	155.00	48.00	1010.00	2120.43	81.00
24	2624.97	111.30	2958.00	97.61	0.10	0.40	9299.75	189.00	0.00	1739.00	2668.80	86.00
25	2388.72	115.41	3023.00	81.97	0.10	0.43	10744.93	160.00	0.00	2387.00	2788.20	90.00
26	2320.35	115.26	3389.00	90.74	0.09	2.48	9821.61	160.00	0.00	2368.00	2826.54	79.00
27	2484.46	117.01	2951.00	91.67	0.10	2.14	10347.60	186.00	10.00	2109.00	3005.95	81.00
28	2131.98	121.21	2232.00	82.67	0.10	0.27	7678.80	153.00	50.00	2046.00	2203.98	80.00
29	2323.86	127.93	3371.00	85.34	0.10	0.35	8454.96	160.00	0.00	2279.00	3281.74	84.00
30	2167.38	121.84	2931.00	86.31	0.09	0.43	8851.32	155.00	0.00	989.00	2690.90	82.00

5.2.3 TYPES OF SIMULATION MODELS

Three different simulation models are presented for the cityscape environment. The first environment was discussed within Section 5.2.1-5.2.2. This section compares the performance of the algorithm across two more cityscapes. These environments have different number of buildings that are placed at random locations. Table 5.4 shows the differences between these models. The simulations that were performed with Cityscape 2 and 3 prioritized lower running time to test the abilities of the algorithm to plan paths across a variety of test spaces.

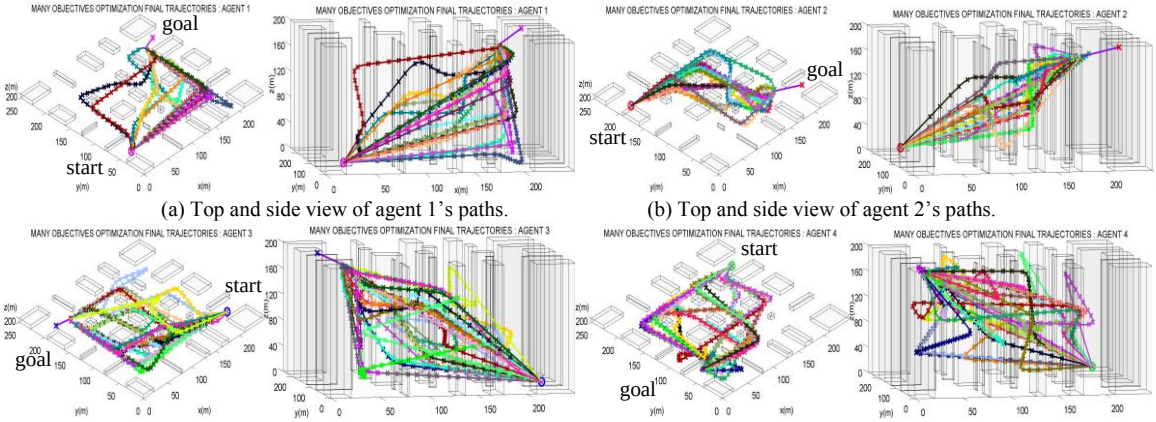
Firstly, in Cityscape 2 and 3, the algorithm is sped up by increasing the maximum distances between two neighbouring path nodes. Only nodes that are more than 20 meters apart are padded with additional nodes between them. This means that the agents will have less stop points across their final trajectories. Secondly, a path in Cityscape 2 and 3 is only sampled once to generate a smooth spline. On the other hand, the paths within Cityscape 1 in Section 5.2.2 was maintained across generations. These paths were resampled at every generation so that the path became smoother at every generation. The data in Figure 5.9-5.10 and Table 5.5-5.6. shows the effects of using low resolution paths with less nodes as opposed to high resolution path with more nodes.

TABLE 5.4: MA-Spread HIGH RISE CITYSCAPE1-3 PARAMETERS.

SIMULATION	HIGH-RISE CITYSCAPE 1	HIGH-RISE CITYSCAPE 2	HIGH-RISE CITYSCAPE 3
RANDOM BUILDINGS	18	25	25
BUILDING SIZE	20m<length<150m	10m<length<50m	8m<length<48m
ROADS	narrow	wide	wide
MAX DISTANCE BETWEEN NODES	2m	20m	20m
SPLINE SAMPLE RATE	5 samples/iteration	5 samples/simulation	5 samples/simulation

The results in Table 5.5 and 5.6 show that all objectives but one is well minimized within both the Cityscape 2 and 3 environments. In fact, six out of the twelve objectives are minimized at a larger percentage than Cityscape 1. The paths in both these environments have less nodes in comparison to Cityscape 1 due to their lower sampling rate. This means that the quadrotors are flying longer distances and any deviation from their smooth paths can accumulate over distances. Thus, the spline deviation cost in Cityscape 2 and 3 isn't easily minimized. On the other hand, Cityscape 1 has a higher sampling rate across generation which reduces the error propagation across two nodes. The control system can minimize the small amount or error across them.

These results show that the algorithm is applicable within a variety of cityscape simulations. The user has the option of increasing the node-to-node sample rate to minimize the spline deviation error. They also have the option of reducing the processing time by reducing the sampling rate. The results within Table 5.5 and 5.6 show that the spline deviation is well within the safety distance of 2 meters between two agents.



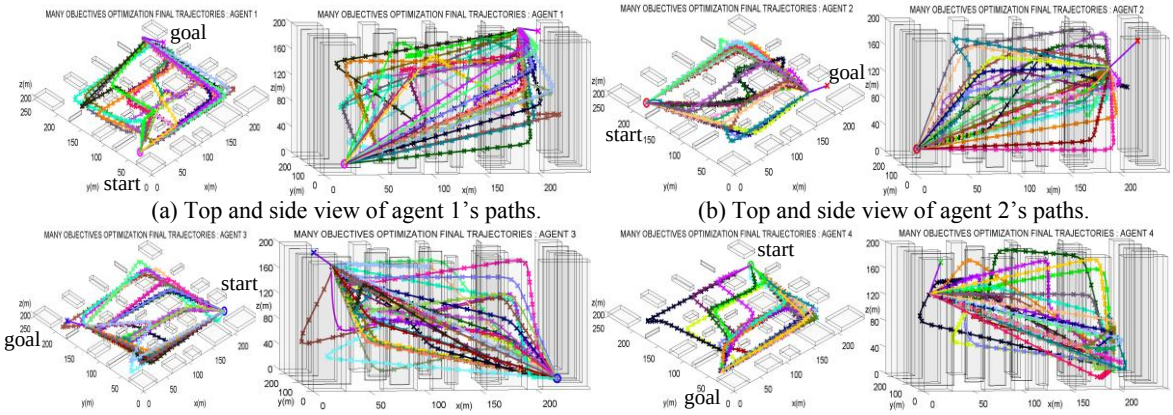
(a) Top and side view of agent 1's paths. (b) Top and side view of agent 2's paths.

(c) Top and side view of agent 3's paths. (d) Top and side view of agent 4's paths.

Fig. 5.9. MA-Spread Cityscape 2: Final generation's multi-agent unique trajectories.

TABLE 5.5: MA-Spread CITYSCAPE 2: MANY-OBJECTIVES OPTIMIZATION RESULTS.

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE (*no free*) *space grids*	SENSORY OVERLAP (*no grid block*) *overlap*
1	max	4862.24	152.71	990.00	95.86	2.22	7.62	2401.31	79.00	38.00	71.00	175.30	220.00
	mean	3154.39	126.79	672.83	75.12	0.93	3.35	1617.19	35.90	6.20	34.10	48.59	157.26
	min	1995.19	97.90	335.00	47.19	0.47	0.87	837.92	10.00	0.00	4.00	3.92	110.00
75	max	1995.45	138.05	320.00	32.90	1.77	3.70	1006.89	29.00	4.00	45.00	55.75	112.00
	mean	1590.42	107.86	241.23	25.93	1.50	0.93	521.86	12.07	0.40	26.80	22.87	84.03
	min	1371.73	71.49	150.00	20.59	1.13	0.30	346.67	9.00	0.00	11.00	0.13	71.00
	%	49.58	14.93	64.15	65.48	-61.02	72.29	67.73	66.38	93.55	21.41	52.93	46.57



(a) Top and side view of agent 1's paths. (b) Top and side view of agent 2's paths.

(c) Top and side view of agent 3's paths. (d) Top and side view of agent 4's paths.

Fig. 5.10. MA-Spread Cityscape 3: Final generation's multi-agent unique trajectories.

TABLE 5.6: MA-Spread CITYSCAPE 3: MANY-OBJECTIVES OPTIMIZATION RESULTS.

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE (*no free*) *space grids*	SENSORY OVERLAP (*no grid block*) *overlap*
1	max	4447.09	144.49	1011.0	106.49	1.37	7.84	2221.89	101.00	32.00	53.00	284.82	218.00
	mean	2894.31	114.45	668.63	72.62	0.84	2.77	1461.17	24.30	2.93	23.47	112.80	141.03
	min	1712.58	85.87	195.00	28.06	0.51	0.00	513.09	10.00	0.00	4.00	7.41	73.00
64	max	1740.61	129.29	259.00	28.34	1.75	3.56	752.16	44.00	8.00	35.00	29.00	83.00
	mean	1576.41	107.30	200.33	25.07	1.53	1.05	466.14	17.20	0.60	15.70	9.43	72.67
	min	1347.41	80.64	100.00	20.54	1.37	0.00	329.23	9.00	0.00	2.00	0.00	55.00
	%	45.53	6.25	70.04	65.48	-82.14	62.07	68.10	29.22	79.52	33.11	91.64	48.47

This path planning and optimization algorithm contains many large and small subsections. Table shows the running time per generation for the for the MA-Spread mission within the cityscape test spaces. Here, the running time for the main subsections is compared to the time it takes to run the GA, MA-RRF Repair and DRMOO collectively. In this case, the smaller subsections of the algorithm are not identified individually.

As previously defined, the paths that were generated across the Cityscape 1 are high resolution. Here, the node-to-node distance is no more than 2 meters. This produces paths that are easily trackable by the quadrotors and have minimal spline deviation error values. This advantage comes at the expense of a longer running time as shown in Table 5.7. Cityscape 2 and 3 that have lower resolution paths have a much shorter running time in comparison to Cityscape 1. These environments also have much lower running times for the multi-agent control system and the calculation of MA-Spread objective values.

Table 5.7 also shows that the GA operators can create two child paths within 5 seconds. It is the MA-RRF path repair process that requires more processing time. One main subsection of the MA-RRF path repair process is the collision check that is required when mapping the free space around obstacles. The collision check is also performed to identify which nodes are redundant within a planned path. In a small sized space, a collision check can run fast due to the minimal number of obstacles. In this study, large spaces with many obstacles will require the algorithm to perform many collision checks for each sample node. These results show how different parameters can affect the error, running time and resolution of the paths across the high-rise cityscape environments. The adjustments of these parameters can be tuned according to the user's preferences.

TABLE 5.7: MA-Spread HIGH RISE CITYSCAPE 1-3 RUNNING TIME.

	ALGORITHM	CITYSCAPE 1 (sec)		CITYSCAPE 2 (sec)		CITYSCAPE 3 (sec)	
MA-SPREAD	GA + MA-RRF Repair + DRMOO	5758.21		913.96		627.77	
	GA	4.973	0.09 %	1.11	0.12 %	0.67	0.11 %
	MA-RRF Repair	1963.49	34.10 %	332.67	36.40 %	155.11	24.71 %
	PD Control System	372.15	6.46 %	129.66	14.19 %	129.06	20.56 %
	Spread objectives	480.96	8.35 %	161.90	17.71 %	144.99	23.10 %
	Collision Check	2867.92	49.81 %	440.00	48.14 %	230.65	36.74 %

5.3. MA-SPREAD ACROSS A HIGHLY CLUTTERED INDOOR ENVIRONMENT

The next simulated environment for the MA-Spread application is the highly cluttered indoor space. This test environment challenges the path planner through its narrow entry points and varying sized clutter across each room. It can be difficult to maintain a diverse set of trajectories because the small number of entry ways between each room. This means that the quadrotors are forced to fly across the same doors and windows as they progress across the test space. There is a large amount of randomly placed clutter across each room.

The path planning algorithm is capable of avoiding the clutter but the agents are once again restricted to flying in similar directions. The results show that the path planning algorithm is capable of minimizing the objectives whilst maintaining a diverse collection of paths.

The results for Indoor 1 are presented in this section. Figures 5.11 and 5.12 shows the progression of the 12 objectives across all generations. Table 5.8 shows the average values of all path combinations for each objective within the final generation. The termination point for the Indoor 1 space is set at the 77th generation. This is the point where most objectives have been sufficiently minimized and path diversity is maintained. Next, the unique paths for each agent are shown in Figure 5.13. These paths will form the path combinations that are shown within Figure 5.14-5.16.

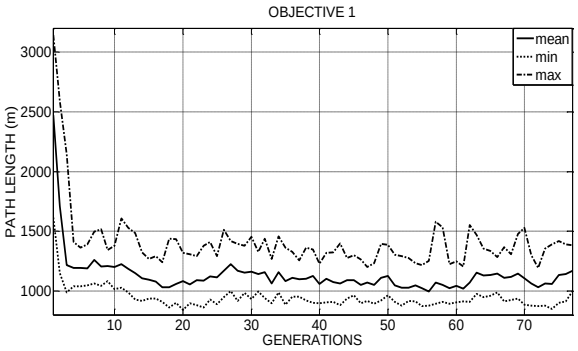
5.3.1 MANY OBJECTIVES VALUES

Figure 5.11 and 5.12 shows the max, mean and min value changes that occur with the 12 objectives across 77 generations for Indoor 1. All objectives have been well minimized. The spline deviation error is maintained which is similar to the cityscape environment. As previously defined, the error is small in value and dependent on the control system. The algorithm is capable of maintaining the deviation error without an increase in value. It also shows that although most objectives are well minimized before the 50th generation, objective 2, 3 and 6 required more time to sufficiently reduce in value.

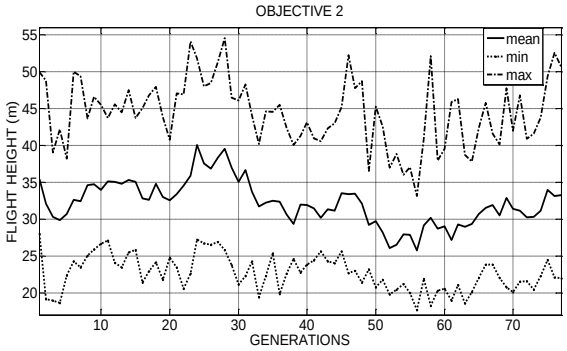
Table 5.8 shows that all the 12 objectives in the final generation are well minimized or maintained. The value for the altitude cost has a small reduction of 3% in comparison to the initial MA-RRF path population. The Indoor 1 environment has a larger number of obstacles in comparison to the cityscape space. These obstacles make it difficult for the paths to only progress across the lower parts of the terrain. The path planning algorithm is also able to generate path nodes that avoid all clutter and walls. Despite this, the number of safety zone breaches is reduced by 38%. The space between rooms is large and can encourage communication network decay. At the 77th generation, the number of broken agent-to-agent connection is decreased by 39%.

All of the other objectives are minimized more than 40% in comparison to the first generation. The path length for each combination is minimized by more than half. It is 53% less in length. Similarly, the number of goal deviations, flight time and jerk cost is minimized by 51%. The agents are able to fly at a speed that minimizes the time optimality cost by 46%. These reductions are difficult to accomplish because there are many path deviations that occur due to the clutter within the environment. The agents must aggressively manoeuvre around each obstacle and this may cause them to fly further away from the goal node. It also requires a large amount of time and fuel to move up and down

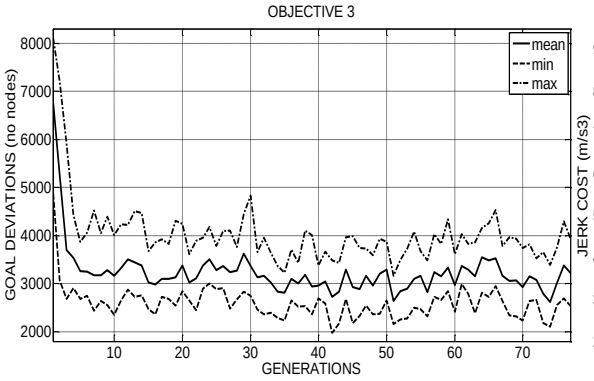
across the clutter. In comparison to the cityscape, there is a larger amount of possible agent-to-agent collisions because the agents are all squeezing through the same entry points. The algorithm is able to reduce this value by 65%. Lastly, the amount of well mapped grid blocks that are visited is reduced by 49%. This means that the agents are visiting more uncertain areas. There is also a reduction of 42% in sensory overlap data. Overall, the algorithm is successful in producing a high percentage of objective value minimization.



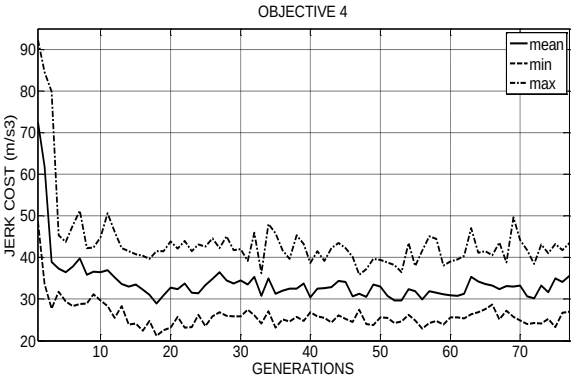
(a) Path length objective values across generations.



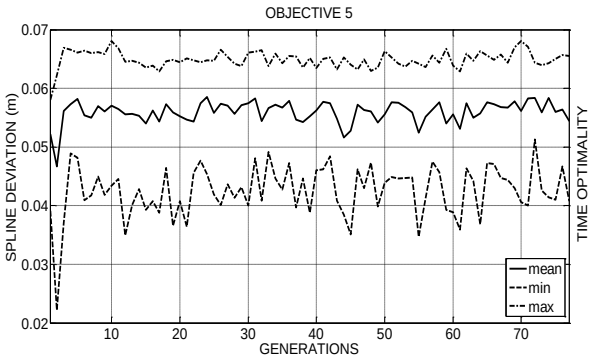
(b) Flight height objective values across generations.



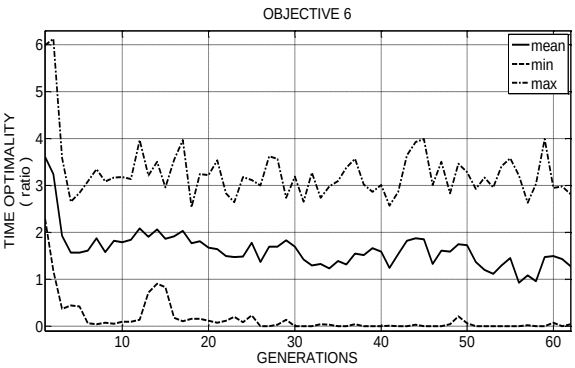
(c) Goal deviations objective values across generations.



(d) Jerk cost objective values across generations.

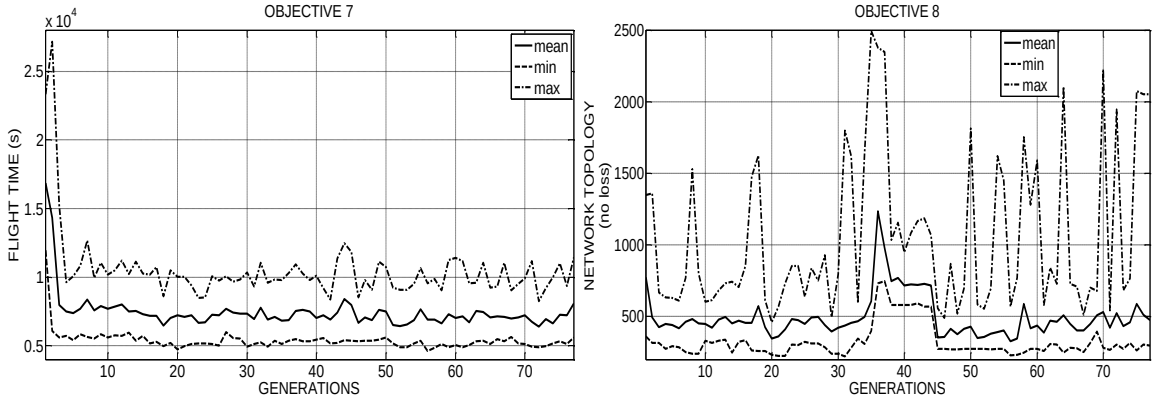


(e) Spline deviation error objective values across generations.

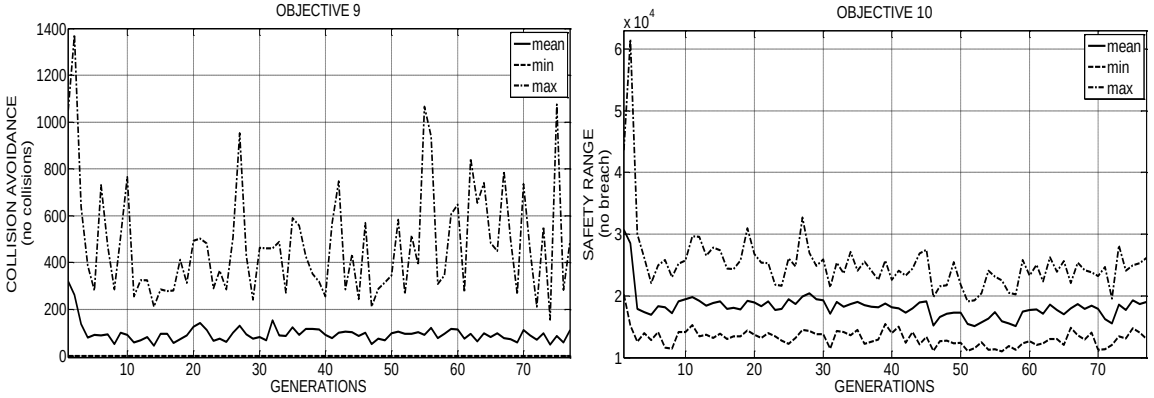


(f) Time optimality objective values across generations.

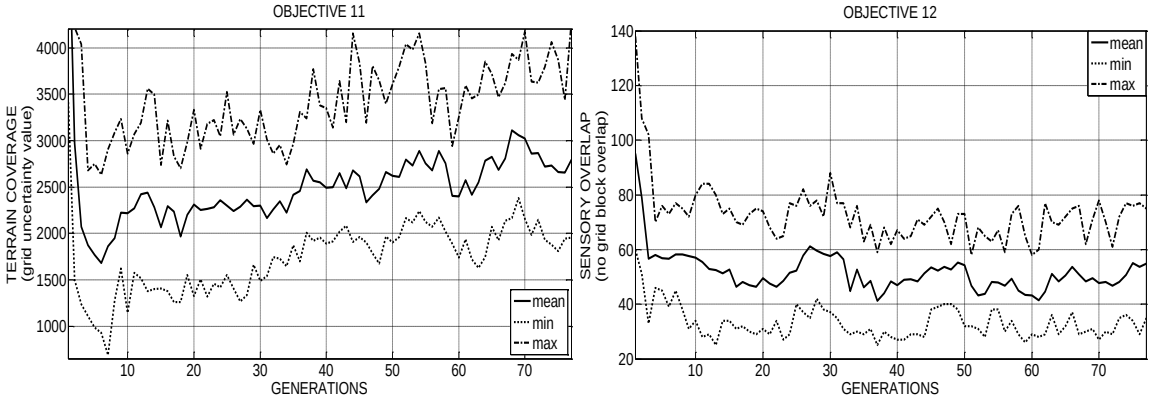
Fig. 5.11. MA-Spread Indoor 1: Progression of the objectives 1-6 across generations.



(a) Flight time objective values across generations. (b) Connection breaches objective values across generations.



(c) Possible collisions objective values across generations. (d)Safety zone breaches objective values across generations.



(e)Uncertain grid coverage objective values across generations. (f)Sensory overlap objective values across generations.

Fig. 5.12. MA-Spread Indoor 1: Progression of the objectives 7-12 across generations.

TABLE 5.8: MA-Spread HIGHLY CLUTTERED INDOOR 1 MANY-OBJECTIVES OPTIMIZATION RESULTS.

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION *(m)*	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE *(no free) space grids*	SENSORY OVERLAP (*no grid block) overlap*
1	max	3142.70	49.93	8104.00	92.32	0.06	5.99	23307.00	1349.00	1050.00	43600.00	6819.70	138.00
	mean	2471.20	35.41	6752.90	72.57	0.05	3.61	16834.00	774.47	320.33	30744.00	5513.30	95.17
	min	1615.70	27.99	4878.00	48.70	0.04	2.29	11940.00	360.00	0.00	20213.00	3413.90	60.00
77	max	1381.00	50.57	3920.00	43.45	0.07	3.79	11445.00	2052.00	494.00	26197.00	4308.10	75.00
	mean	1167.70	33.28	3216.50	35.61	0.05	1.96	8067.80	475.87	111.33	19017.00	2794.40	54.77
	min	1000.20	21.95	2528.00	26.95	0.04	0.00	5599.70	296.00	0.00	12986.00	1953.30	35.00
	%	52.75	6.02	52.37	50.93	0.00	45.71	52.07	38.56	65.25	38.14	49.32	42.45

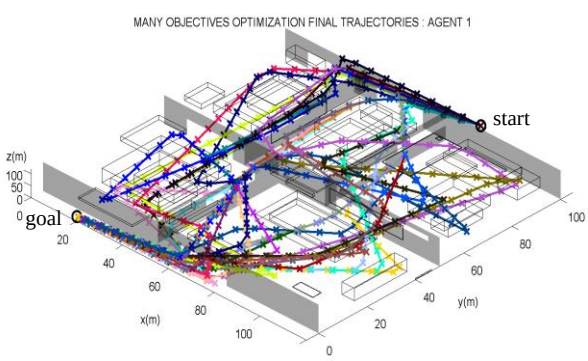
5.3.2 TRAJECTORY POPULATION

The unique paths for each agent within the 77th generation for Indoor 1 are shown in Figure 5.13. Each unique trajectory is displayed in a different colour. The paths for all agents are diverse in nature with the exception of agent 2. The trajectories for agent 2 have many nodes across the lower part of the Indoor 1 space. Still, Figure 5.13(b) shows that there are paths that span across each room.

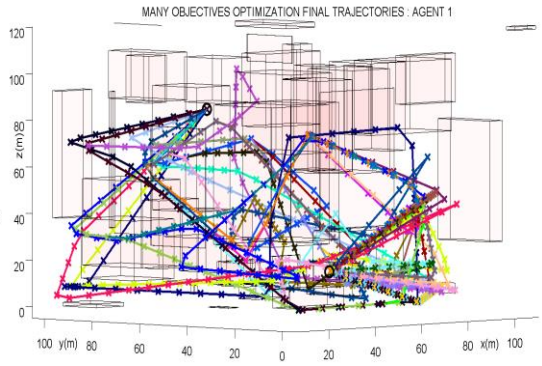
The final population of path combinations is presented in Figure 5.14-5.16. These solutions are made up of the best combinations of each agent's unique paths. The paths of the four quadrotor agents within the Indoor 1 environment are marked in brown, black, blue and green. The first combination holds paths with the least number of possible agent-to-agent collisions. It also has the minimal value for time optimality as well as sensory data overlaps. Figure 5.14(a) shows that the paths for each agent are well spread and allow to agents to maintain a safe distance from each other. This also means that the agents collect sensory data from different parts of the environment. Figure 5.14(b) shows that combination 2 allows the agents to effectively fly across obstacles with the least number of safety zone breaches. The agents fly either above or below the clutter within each room.

Option 10 is another great choice for the end user. It has minimal costs for the number of goal node deviations, jerk cost and flight time. Figure 5.14(j) shows that many of the agents fly directly towards their goal nodes. Combination 12 is the best choice if the end user requires minimal network decay between all agents during a mission. Figure 5.15(b) shows that the agents are always within close proximity of each other. It can be seen in Figure 5.15(e) that option 15 produces paths at a lower altitude. This combination has the minimal cost in terms of flight height since many of the nodes are located at the lower parts of the environment. Combination 16 has the maximum cost for goal node deviation. Figure 5.53(f) shows that the path in blue for agent 3 travels a large distance away from the goal node before redirecting itself towards it. The most common goal for end users is to determine which combination produces the shortest paths.

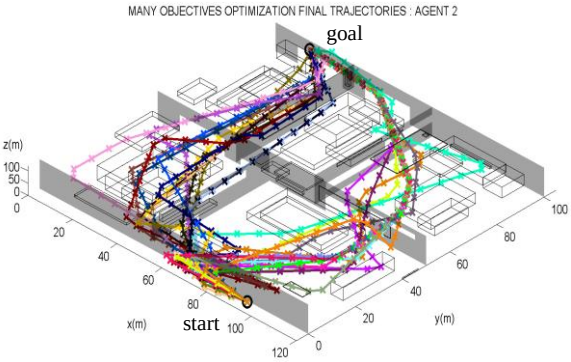
Figure 5.16(j) shows that combination 20 provides the paths with minimal travel distance. Three out of the four agent travel directly from the start to goal node. On the other hand, option 23 contains the paths with the most length. This would be an unfavourable choice for an end user that prioritises minimal flight time. Combination 29 offers the end user paths that travel across uncertain areas of the environment. This option allows the quadrotors to fly across areas that are least mapped within the cluttered rooms. Lastly, option 30 is the path with the most fuel consumption. It also produces paths that require the agents to fly at a less optimal speed. Figure 5.16(j) shows that the paths require the agents to perform many turns and travel further distances.



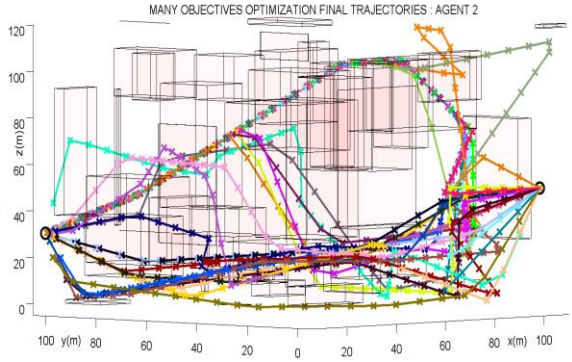
(a) Top view of agent 1's paths within the final generation.



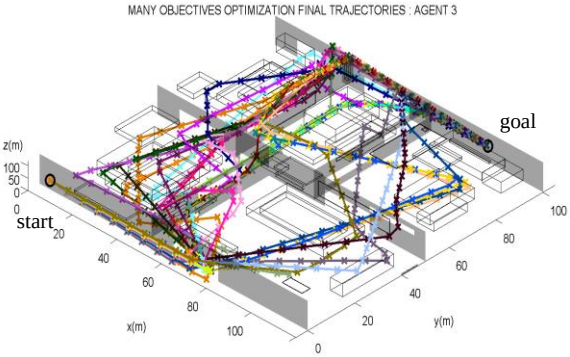
(b) Side view of agent 1's paths within the final generation.



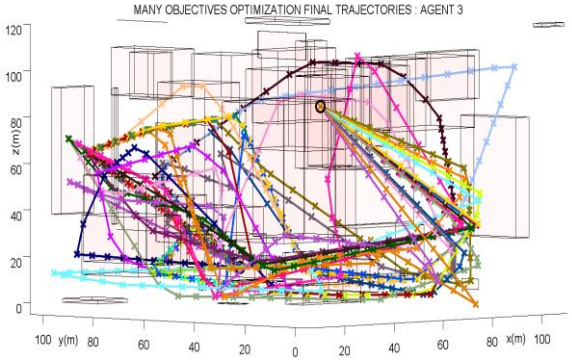
(c) Top view of agent 2's paths within the final generation.



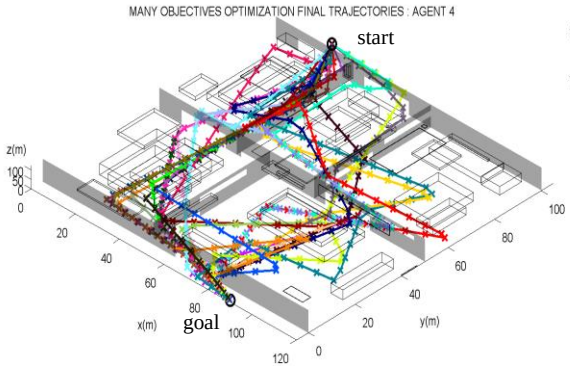
(d) Side view of agent 2's paths within the final generation.



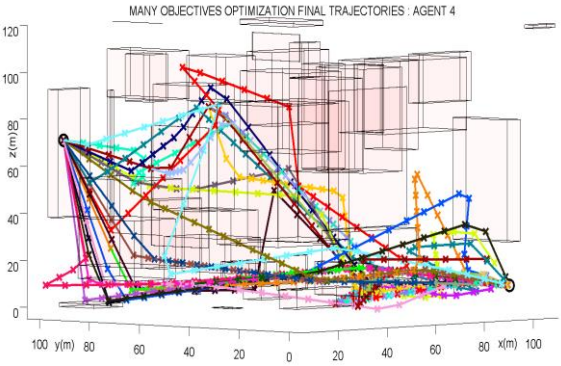
(e) Top view of agent 3's paths within the final generation.



(f) Side view of agent 3's paths within the final generation.

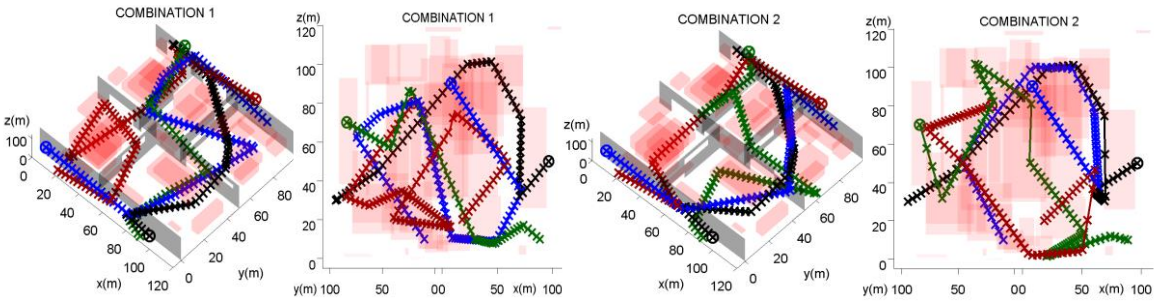


(g) Top view of agent 4's paths within the final generation.



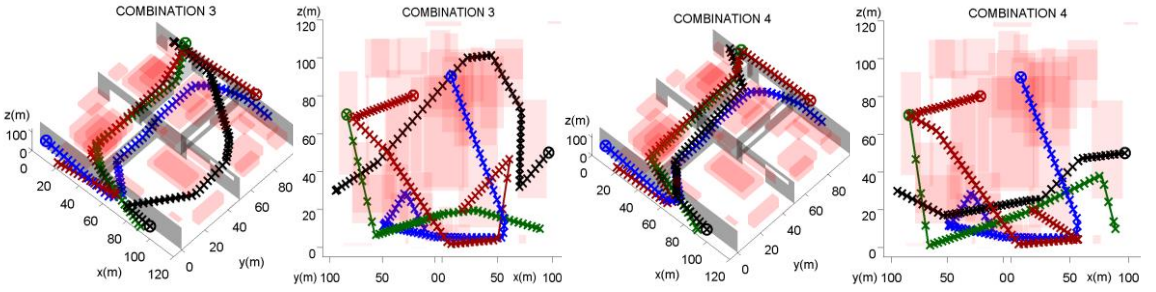
(h) Side view of agent 4's paths within the final generation.

Fig. 5.13. MA-Spread Indoor 1: Final generation's multi-agent unique trajectories.



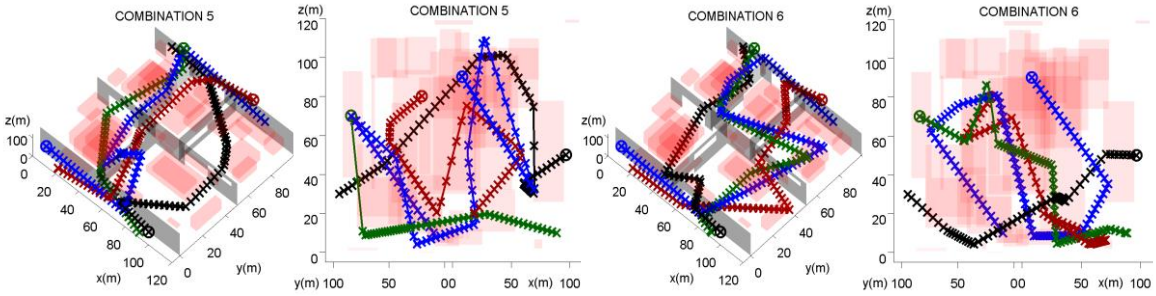
(a) Multi-agent trajectories within combination 1.

(b) Multi-agent trajectories within combination 2.



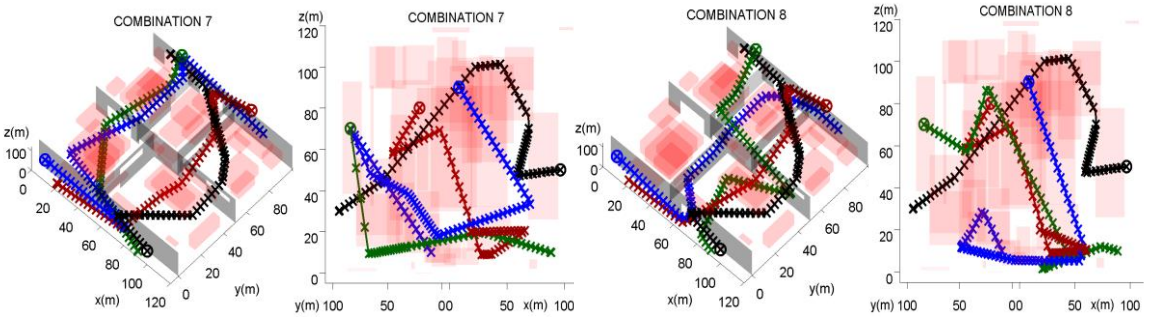
(c) Multi-agent trajectories within combination 3.

(d) Multi-agent trajectories within combination 4.



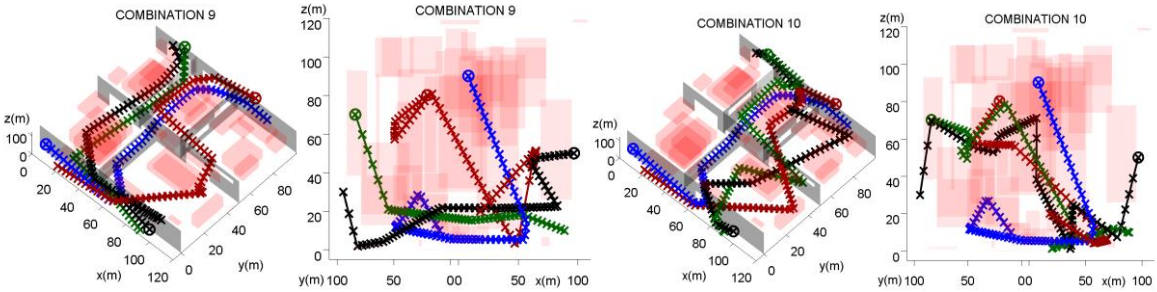
(e) Multi-agent trajectories within combination 5.

(f) Multi-agent trajectories within combination 6.



(g) Multi-agent trajectories within combination 7.

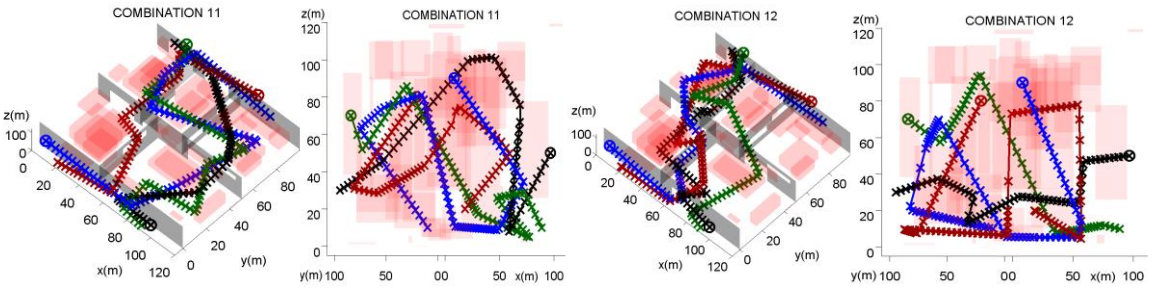
(h) Multi-agent trajectories within combination 8.



(i) Multi-agent trajectories within combination 9.

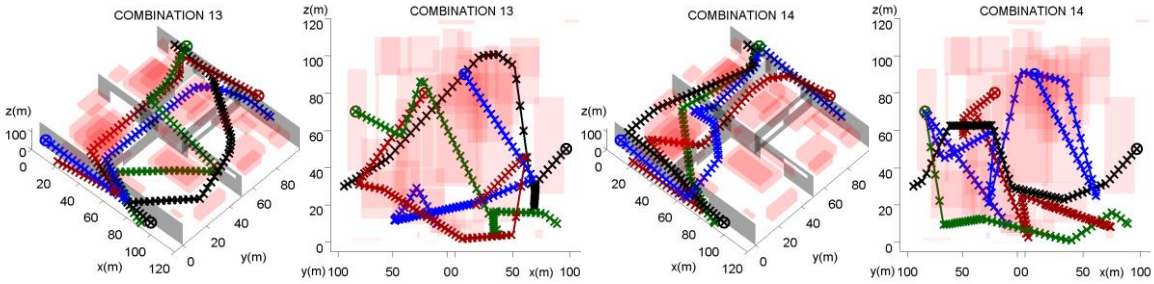
(j) Multi-agent trajectories within combination 10.

Fig. 5.14. MA-Spread Indoor 1: Multi-agent path combination 1-10 for the final generation.



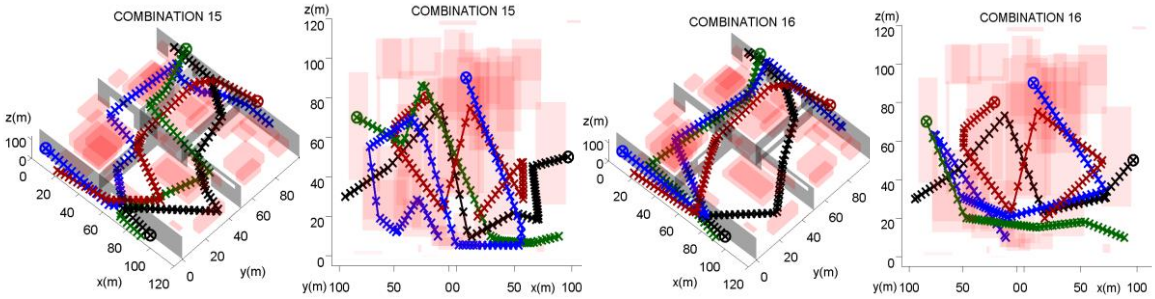
(a) Multi-agent trajectories within combination 11.

(b) Multi-agent trajectories within combination 12.



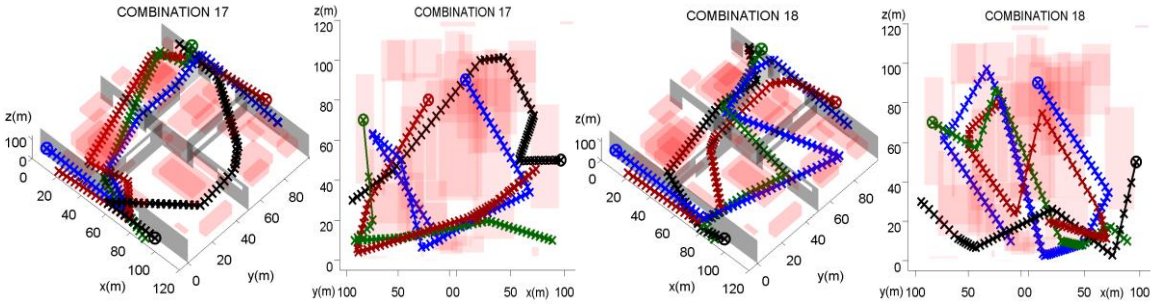
(c) Multi-agent trajectories within combination 13.

(d) Multi-agent trajectories within combination 14.



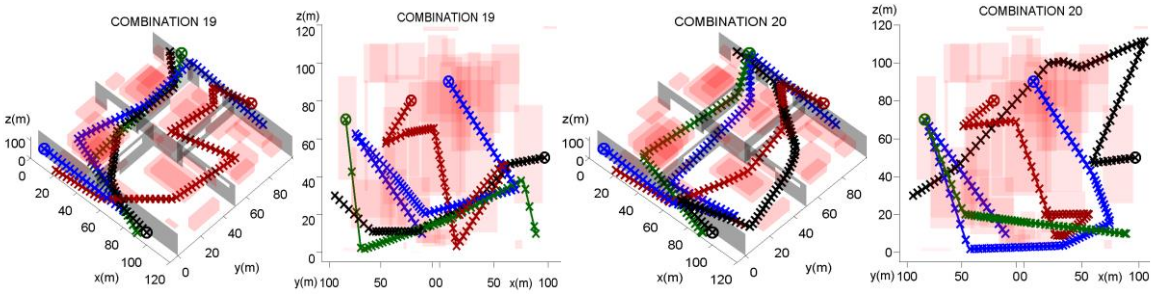
(e) Multi-agent trajectories within combination 15.

(f) Multi-agent trajectories within combination 16.



(g) Multi-agent trajectories within combination 17.

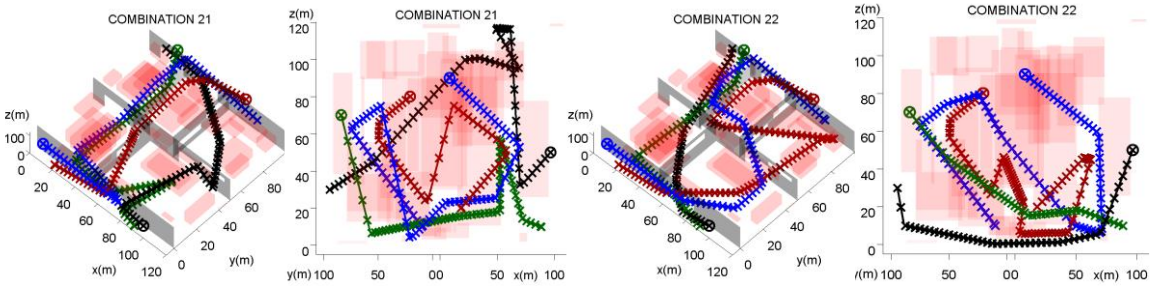
(h) Multi-agent trajectories within combination 18.



(i) Multi-agent trajectories within combination 19.

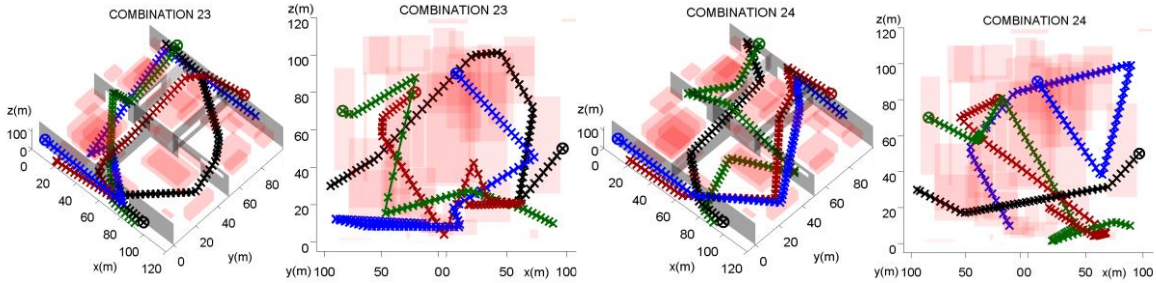
(j) Multi-agent trajectories within combination 20.

Fig. 5.15. MA-Spread Indoor 1: Multi-agent path combination 11-20 for the final generation.



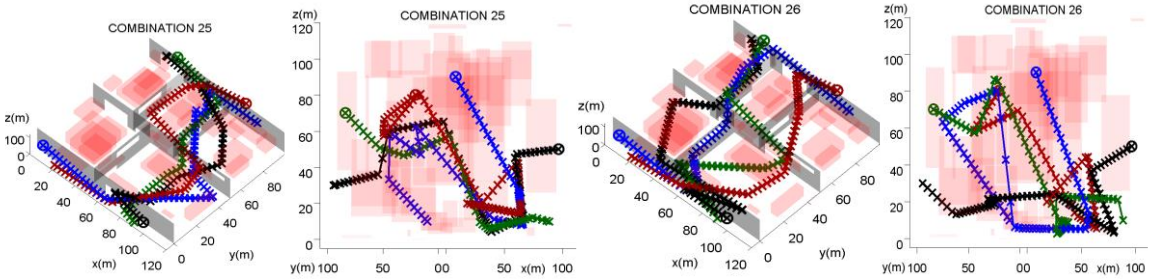
(a) Multi-agent trajectories within combination 21.

(b) Multi-agent trajectories within combination 22.



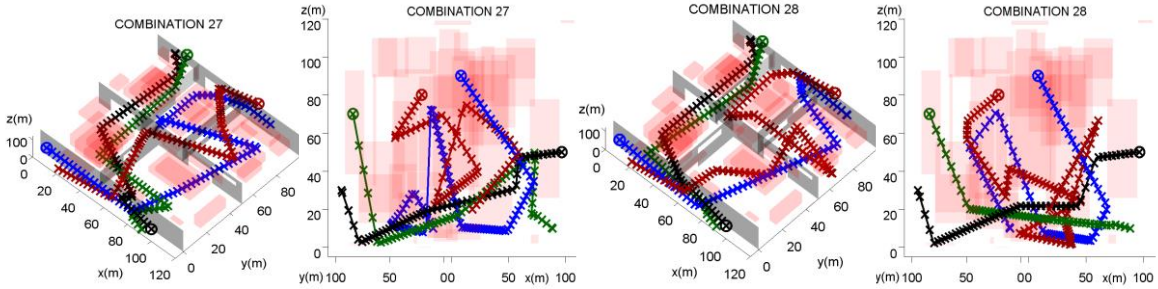
(c) Multi-agent trajectories within combination 23.

(d) Multi-agent trajectories within combination 24.



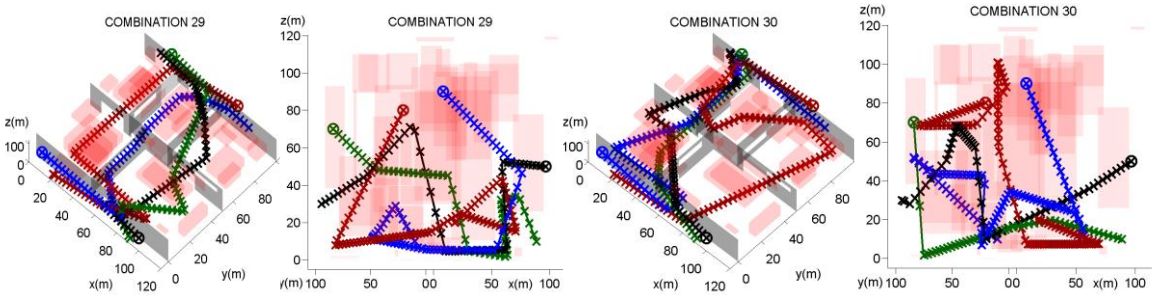
(e) Multi-agent trajectories within combination 25.

(f) Multi-agent trajectories within combination 26.



(g) Multi-agent trajectories within combination 27.

(h) Multi-agent trajectories within combination 28.



(i) Multi-agent trajectories within combination 29.

(j) Multi-agent trajectories within combination 30.

Fig. 5.16. MA-Spread Indoor 1: Multi-agent path combination 21-30 for the final generation.

TABLE 5.9: MA-Spread HIGHLY CLUTTERED INDOOR FINAL GENERATION'S RESULTS.

INDOOR 1 Spread COMBINATION	PATH LENGTH (*meters*)	ALTITUDE COST (*meters*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE ERROR *(m)*	TIME OPTIMALITY *(ratio)*	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE (*% free space*)	SENSORY DATA OVERLAP (*grid block overlap*)
1	1085.39	25.46	2812.00	32.69	0.06	0.00	6142.79	449.00	0.00	19387.00	2486.97	35.00
2	1307.61	41.94	3219.00	35.30	0.06	1.76	7153.15	302.00	86.00	12986.00	2154.57	66.00
3	1294.82	50.57	2823.00	39.62	0.07	1.55	6156.02	333.00	182.00	18346.00	2237.28	69.00
4	1175.48	37.76	2986.00	30.05	0.06	1.56	6137.03	302.00	0.00	19138.00	2389.77	57.00
5	1203.98	39.71	2852.00	39.40	0.06	2.04	7548.68	362.00	212.00	16623.00	2405.72	62.00
6	1177.20	34.54	2745.00	32.41	0.06	1.68	6431.61	333.00	0.00	16357.00	2051.55	60.00
7	1030.11	25.86	2955.00	33.23	0.05	1.98	7595.15	378.00	104.00	18041.00	2954.37	48.00
8	1296.04	34.83	3611.00	36.81	0.06	2.05	8359.28	502.00	138.00	20248.00	2571.84	57.00
9	1354.69	39.10	3448.00	43.45	0.06	2.13	7886.36	351.00	114.00	18861.00	2909.62	75.00
10	1079.67	45.29	2528.00	26.95	0.06	1.09	5599.68	454.00	0.00	17030.00	3026.33	52.00
11	1141.77	28.30	3469.00	39.31	0.05	2.31	9080.41	821.00	0.00	23208.00	2492.55	61.00
12	1225.28	42.75	2927.00	35.01	0.06	1.40	6250.70	296.00	0.00	17162.00	1981.50	74.00
13	1193.03	33.20	3042.00	35.71	0.06	0.14	6898.99	305.00	98.00	16176.00	3065.71	63.00
14	1312.48	28.62	3602.00	36.33	0.05	1.89	8493.41	647.00	114.00	18781.00	3474.56	52.00
15	1163.66	21.95	2997.00	40.44	0.05	1.62	9758.43	333.00	136.00	22119.00	2329.10	57.00
16	1135.44	24.25	3920.00	31.99	0.06	2.54	8329.20	389.00	346.00	14520.00	4308.08	39.00
17	1053.53	33.04	3007.00	32.61	0.06	0.56	7294.01	388.00	150.00	13163.00	3088.96	54.00
18	1156.17	35.58	3072.00	34.02	0.06	1.70	6428.79	2052.00	220.00	19749.00	2450.70	46.00
19	1061.08	28.36	3237.00	34.23	0.05	1.72	9336.19	304.00	52.00	18136.00	3475.99	44.00
20	1000.19	34.39	3094.00	34.49	0.05	2.56	7466.83	388.00	494.00	22386.00	3100.97	46.00
21	1289.80	30.68	3458.00	42.46	0.05	2.73	9933.63	671.00	246.00	23122.00	3863.67	46.00
22	1216.31	31.72	3591.00	35.50	0.05	2.73	9515.10	567.00	0.00	19438.00	3449.72	58.00
23	1381.01	31.53	3386.00	39.29	0.06	2.23	8998.37	296.00	126.00	17870.00	2245.42	66.00
24	1151.37	38.26	3230.00	36.45	0.05	2.24	7760.31	296.00	96.00	20409.00	3109.21	52.00
25	1087.80	31.88	3695.00	35.40	0.05	2.93	9371.13	304.00	0.00	20294.00	2784.12	48.00
26	1084.84	28.77	3162.00	33.22	0.05	1.68	7486.46	399.00	108.00	15716.00	3046.03	51.00
27	1095.03	26.74	3766.00	36.59	0.04	2.45	9455.15	320.00	204.00	18036.00	3044.83	57.00
28	1082.19	32.22	3181.00	31.66	0.05	2.67	8954.63	514.00	114.00	21135.00	2730.27	45.00
29	1107.19	33.69	3215.00	38.79	0.04	3.18	10768.54	656.00	0.00	26197.00	1953.30	59.00
30	1088.22	27.41	3464.00	34.91	0.04	3.79	11445.30	564.00	0.00	25882.00	2648.52	44.00

This shows that even well minimized path combinations have their own disadvantages. As with the cityscape environment, all of the combinations within the Indoor 1 space are well minimized and have a variety of trade off values. The end user is able to use their post processing preferences in order to determine the best trajectory combination for their mission.

5.3.3 TYPES OF SIMULATION MODELS

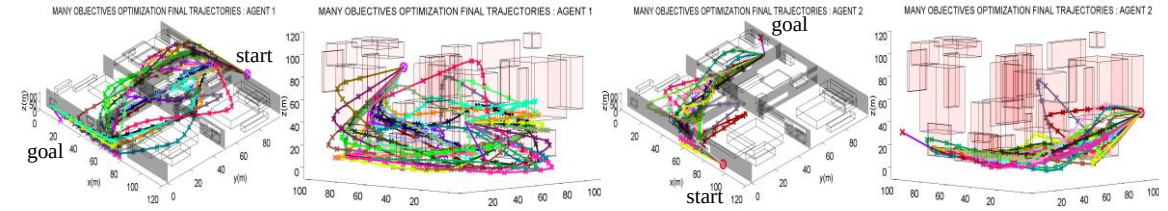
Like the cityscape environment, three different simulation models are presented for the indoor environment. The first environment was discussed within Section 5.3.1-5.3.2. This section compares the performance of the algorithm across two more indoor test spaces. These environments vary in number of randomly placed clutter per room. Table 5.10 shows the differences between these models. The algorithm is sped up by increasing the maximum distances between two neighbouring path nodes within the Indoor 2 and 3. Only nodes that are more than 20 meters apart are padded with additional nodes between them. Also, a path in the Indoor 2 and 3 spaces is only sampled once to generate a smooth spline.

TABLE 5.10: MA-Spread HIGHLY CLUTTERED INDOOR 1-3 PARAMETERS.

SIMULATION	HIGHLY CLUTTERED INDOOR 1	HIGHLY CLUTTERED INDOOR 2	HIGHLY CLUTTERED INDOOR 3
RANDOM CLUTTER	8	6	4
CLUTTER SIZE	1m<length<35m	2m<length<35m	2m<length<30m
MAX DISTANCE BETWEEN NODES	2m	20m	20m
SPLINE SAMPLE RATE	5 samples/iteration	5 samples/simulation	5 samples/simulation

The results of Indoor 2 in Table 5.11 -5.12 show that all the objectives except three are minimized at a larger percentage in comparison to Indoor 1. Similarly, all the objectives within Indoor 3 are minimized at a larger percentage in comparison to Indoor 1. The largest difference between the three environments is the increase in the spline deviation error across generations. This issue exists for the same reason that it occurs in Cityscape 2 and 3. These paths are designed with larger node-to-node distances which can lead to larger error propagation. The agents still do not deviate more than the minimum neighbour-to-neighbour distance of 2 meters.

The data for all three environments shows that the algorithm can perform well within the different variations of the indoor space. It can avoid different amounts of randomly placed clutter that vary in size. The user has the option of running a longer yet high resolution path planner as shown in Indoor 1. They also can run a much faster version of the algorithm with lower resolution paths as designed in Indoor 2 and 3 as shown in Table 5.13. The completion time for the algorithm across all three indoor variations



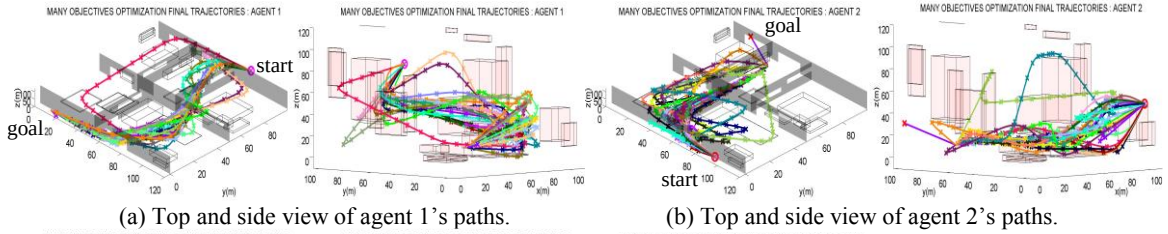
(a) Top and side view of agent 1's paths. (b) Top and side view of agent 2's paths.

(c) Top and side view of agent 3's paths. (d) Top and side view of agent 4's paths.

Fig. 5.17. MA-Spread Indoor 2: Final generation's multi-agent unique trajectories.

TABLE 5.11: MA-Spread INDOOR 2: MANY-OBJECTIVES OPTIMIZATION RESULTS.

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE (*no free) space grids*	SENSORY OVERLAP (*no grid block) overlap*
1	max	3017.45	59.40	838.00	73.51	1.68	8.18	2416.35	51.00	786.00	6104.00	383.89	121.00
	mean	1785.07	42.11	554.17	54.35	0.68	3.90	1425.83	27.17	84.07	2327.50	192.85	85.73
	min	938.45	24.90	156.00	21.03	0.33	1.14	308.02	12.00	0.00	354.00	51.16	37.00
60	max	1169.44	39.66	226.00	22.96	1.84	3.59	1017.14	38.00	64.00	1131.00	178.67	47.00
	mean	979.45	30.77	177.60	19.63	1.51	1.46	391.28	19.73	25.13	697.53	113.84	37.00
	min	858.65	20.91	139.00	15.93	1.19	0.00	240.18	10.00	0.00	413.00	53.22	26.00
	%	45.13	26.93	67.95	63.88	-122.06	62.57	72.56	27.38	70.11	70.03	40.97	56.84



(a) Top and side view of agent 1's paths. (b) Top and side view of agent 2's paths.

(c) Top and side view of agent 3's paths. (d) Top and side view of agent 4's paths.

Fig. 5.17. MA-Spread Indoor 3: Final generation's multi-agent unique trajectories.

TABLE 5.12: MA-Spread INDOOR 3: MANY-OBJECTIVES OPTIMIZATION RESULTS.

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE (*no free) space grids*	SENSORY OVERLAP (*no grid block) overlap*
1	max	2940.20	54.43	1100.00	91.18	0.78	6.45	2700.72	570.00	404.00	3738.00	478.13	121.00
	mean	2116.15	40.58	750.50	68.74	0.55	3.66	1805.09	61.23	36.07	2107.37	235.62	90.20
	min	1404.37	20.65	422.00	47.71	0.37	2.14	832.33	17.00	0.00	747.00	66.13	59.00
60	max	1327.94	49.91	238.00	28.54	1.73	4.11	708.55	30.00	40.00	1023.00	118.83	57.00
	mean	972.29	36.27	187.10	20.74	1.36	1.73	404.08	17.27	13.73	639.83	57.58	39.43
	min	848.65	27.10	147.00	15.86	1.01	0.54	251.64	11.00	0.00	434.00	20.17	26.00
	%	54.05	10.62	75.07	69.83	-147.27	52.73	77.61	71.79	61.94	69.64	75.56	56.29

are faster than the cityscape environments because it is smaller in size and contains more free space. It also shows similarities with the cityscape environments where the collision check process requires the most processing time. This is because for each MA-RRF repair node, the algorithm must test if the sample node collides with any obstacle. The indoor environment contains many clutter and entryways that need to be considered.

The percentage of processing time for the indoor environments is different for all three variations. Indoor 1 uses a significant amount of time on calculating the MA-Spread objectives. On the other hand, the control system within the Indoor 2 simulation uses a larger percentage of the total time. Lastly, Indoor 3 requires the highest amount of time for the child path repair process.

The results for the indoor environment show how different values for the variables can affect the error, running time and resolution of the paths. These variables can be tuned according to the user's preferences.

TABLE 5.13: MA-Spread HIGHLY CLUTTERED INDOOR 1-3 RUNNING TIME.

	ALGORITHM	INDOOR 1 (sec)		INDOOR 2 (sec)		INDOOR 3 (sec)	
MA-SPREAD	GA + MA-RRF Repair + DRMOO	2095.60		735.51		792.88	
	GA	0.62	0.03 %	0.72	0.10 %	0.70	0.09 %
	MA-RRF Repair	153.41	7.32 %	207.84	28.56 %	374.28	47.21 %
	PD Control System	281.50	13.43 %	156.10	21.22 %	93.12	11.74 %
	Spread objectives	504.62	24.08 %	65.86	8.95 %	137.55	17.35 %
	Collision Check	591.78	28.24 %	426.55	57.99 %	584.79	73.76 %

5.4. MA-SPREAD ACROSS A MOUNTAINOUS TERRAIN

Lastly, the mountainous terrain brings different challenges to the MA-Spread application. The final environment has simplistic trajectories due to its large amount of free space between peaks. It is different from the cityscape and indoor environment that maintains uniform side planes. It can be challenging because it requires accurate high resolution free space mapping in order to avoid collisions between the agents and the mountain peaks. Here, the quadrotors are required to fly through sudden height changes in the terrain. The changes in height are in the form of mountain peaks and lows. This means that the environment is split into small sized grid blocks in order to accurately map the constant height changes. Thus, this environment holds a huge number of small sized obstacle blocks.

Similar to the prior environments, the different trade-off values for the 12 objective functions are evaluated in this section. Firstly, the progression of the objectives across generations for Mountain 1 is shown in Figure 5.18-5.19. Then, the minimization of the cost functions for the entire population in the final generation is shown in Table 5.14. The unique paths for all four agents are presented in Figure 5.20. Lastly, the imagery of each

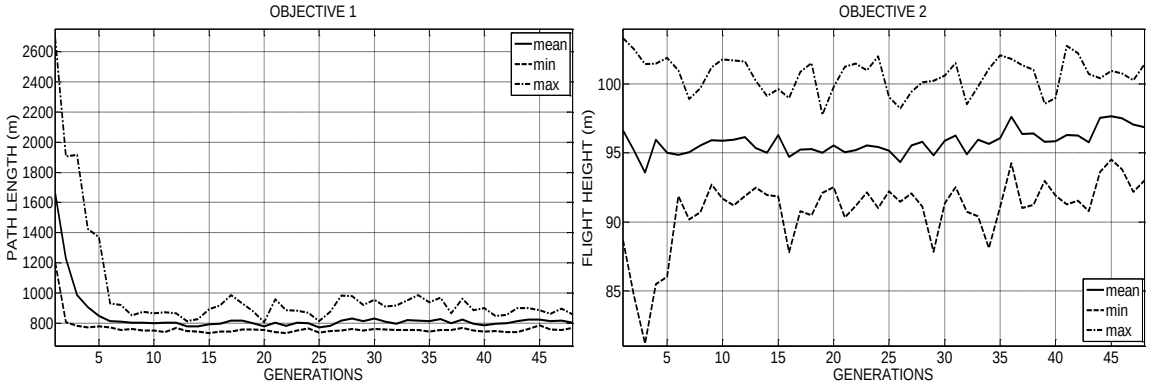
path combination within the final population is shown in Figure 5.21-5.23. The objective trade-off values for each combination are shown in Table 5.15.

5.4.1 MANY OBJECTIVES VALUES

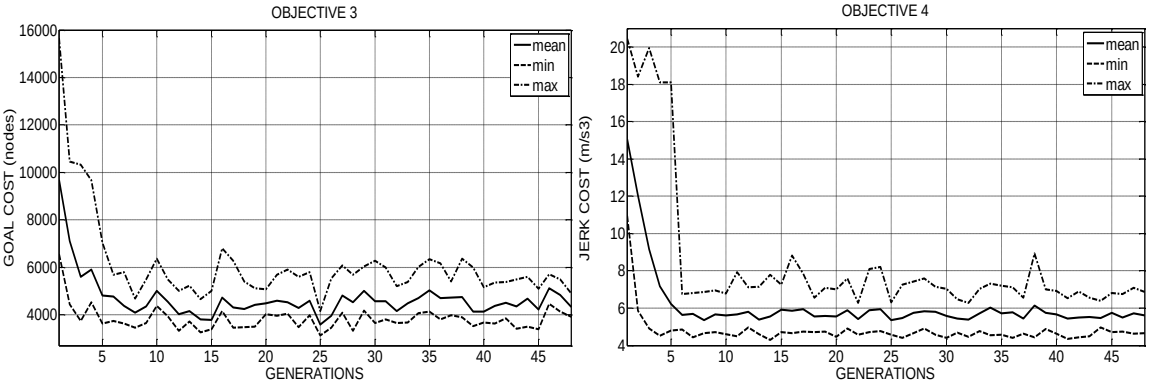
The variations of the values for each objective in Mountain 1 are shown in Figure 5.18 and 5.19. The 12 objectives within this environment reach their minimal value at a faster rate in comparison to the other test spaces. This could be due to the noncomplex trajectories and large amount of free space within the terrain. Most objectives reach their minimum points within 40 generations. The only exception to this is objective 7 and 10. The termination point for the optimization algorithm is set at the 48^{th} iteration. Here as well, the graphs show that all the objectives are not degraded across generations. They are either minimized or maintained in comparison to the first generation.

Similar to the last two test spaces; Table 5.14 shows that all the objectives in the final generation are well minimized or maintained. It shows that two objectives have been maintained across the 48 generations. These are the altitude and spline deviation cost. Three cost functions are minimized by less than 40% reduction in value. The number of agent-to-agent connection loss is lowered by 29%. Flights across well mapped areas are decreased by 26% whereas the number of sensory overlap data is reduced by 19%.

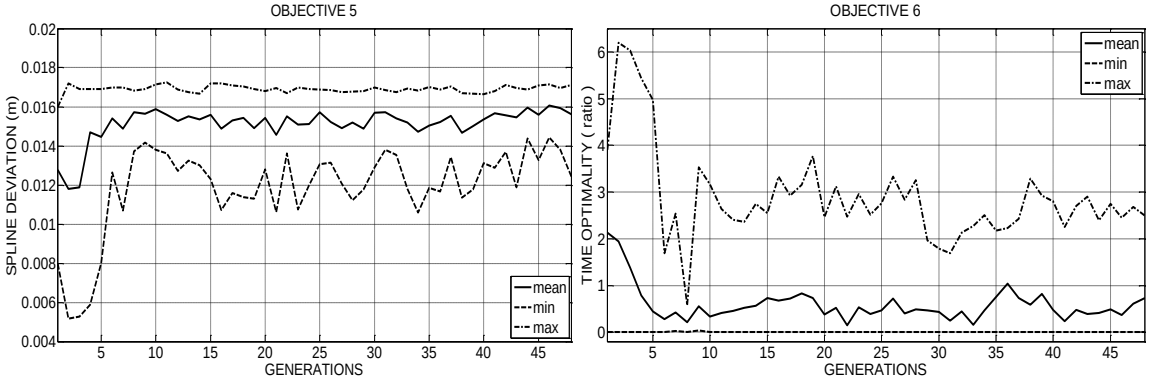
All of the other objectives are minimized by more than 40%. The path lengths for all four quadrotors are lowered by 52% in comparison to the initial MA-RRF paths. Next, the number of goal node deviations is also minimized by 55%. The mountainous terrain doesn’t require the agents to perform many sharp turns. Thus, the jerk cost for the team of agents is reduced by 63%. Similarly, the time optimality cost is lessened by 65%. The shorter and more direct paths lead to a 60% reduction of fuel consumption. The number of agent-to-agent possible collisions is minimized by 45%. Lastly, the number of safety zone breaches is reduced by 57% despite the large number of obstacles within the Mountain 1 terrain.



(a) Path length objective function across generations. (b) Flight height objective values across generations.

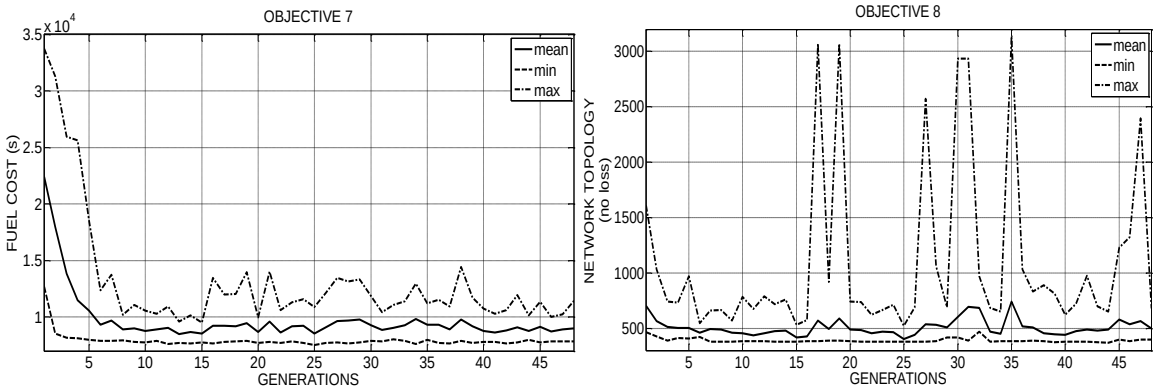


(c) Goal deviations objective values across generations. (d) Jerk cost objective values across generations.

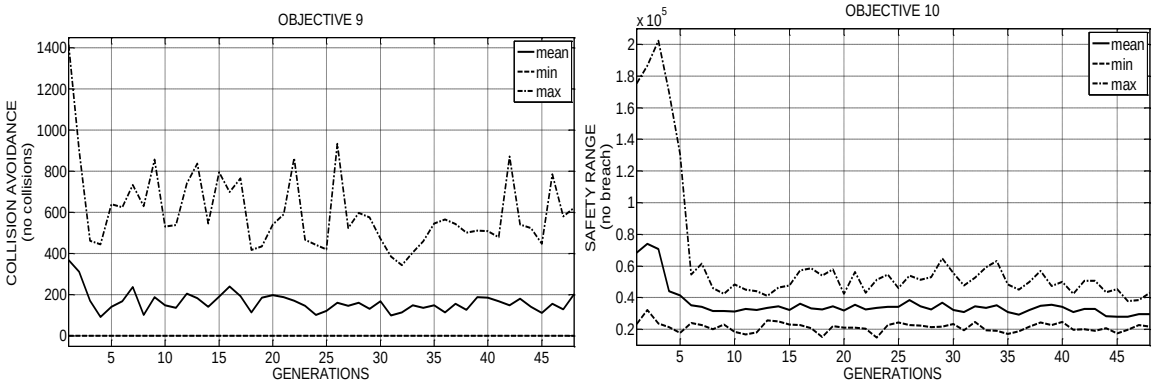


(e) Spline deviation error objective values across generations. (f) Time optimality objective values across generations.

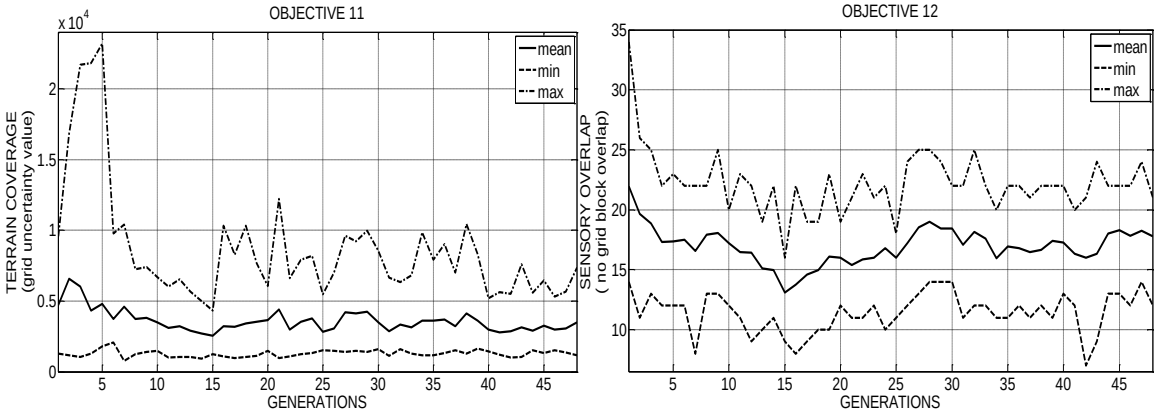
Fig. 5.18. MA-Spread Mountain 1: Progression of the objectives 1-2 across generations.



(a) Flight time objective values across generations. (b) Connection breaches objective values across generations.



(c) Possible collisions objective values across generations.(d) Safety zone breaches objective values across generations.



(e)Uncertain grid coverage objective values across generations. (f)Sensory overlap objective values across generations.

Fig. 5.19. MA-Spread Mountain 1: Progression of the objectives 7-12 across generations.

TABLE 5.14: MA-Spread MOUNTAIN TERRAIN 1 MANY-OBJECTIVES OPTIMIZATION RESULTS.

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION *(m)*	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE *(no free) space grids*	SENSORY OVERLAP (*no grid block) overlap*
1	max	2691.70	103.34	15665.00	20.47	0.02	3.90	33746.00	1611.00	1414.00	175256.00	9591.20	34.00
	mean	1664.50	96.63	9692.00	15.07	0.01	2.14	22413.00	703.13	369.00	68478.00	4729.10	21.93
	min	1204.90	88.68	6559.00	10.96	0.00	0.00	12674.00	464.00	0.00	23420.00	1287.10	14.00
48	max	860.49	101.44	4889.00	6.88	0.02	2.50	11465.00	678.00	622.00	43183.00	7356.30	21.00
	mean	803.32	96.87	4320.80	5.59	0.02	0.74	8980.60	497.23	202.33	29513.00	3479.50	17.77
	min	769.36	93.02	3911.00	4.66	0.01	0.00	7837.60	399.00	0.00	21756.00	1169.70	12.00
	%	51.74	-0.25	55.42	62.91	-100.00	65.42	59.93	29.28	45.17	56.90	26.42	18.97

5.4.2 TRAJECTORY POPULATION

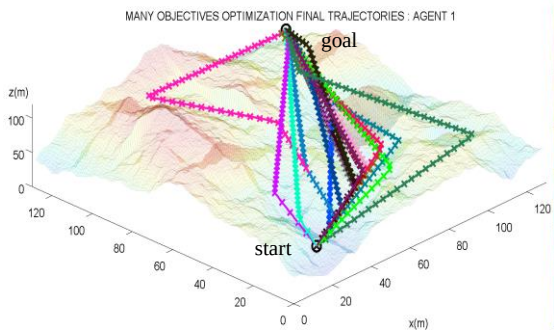
The path combinations within the final population for Mountain 1 are made up of each agent's unique paths. These paths are presented in Figure 5.20. Each path is defined by different colours. It is difficult to maintain a certain level of diversity within the mountainous terrain because the mountains are spread across large areas. The free spaces between the mountain bases are also large. The paths in each generation are reduced in size through redundant node pruning. Thus, the diverse paths are reduced to nodes that span across similar areas. The cityscape and indoor test space have many smaller sized obstacles within the environment. Thus, the solution population after the path pruning process is still diverse in nature.

Next, the path combinations within the final generation are shown in Figure 5.21-5.23. The paths for all four agents are shown within its imagery with nodes marked in black, blue, green and brown. Combination 1 produces paths that have the minimum value for flight time. Figure 5.21(a) shows that the paths are simplistic and easy to track. It also allows the agents to fly faster than the average velocity. This leads to the minimum value for time optimality. The choice with the least number of possible agent-to-agent collisions is combination 2. Similarly, combination 5 produces paths with the lowest amount of safety zone breaches. These two options are preferable for the end user that requires paths with little replanning. Combination 10 has many advantages such as providing the minimum value for number of goal node deviations, uncertain space coverage and sensory data overlap.

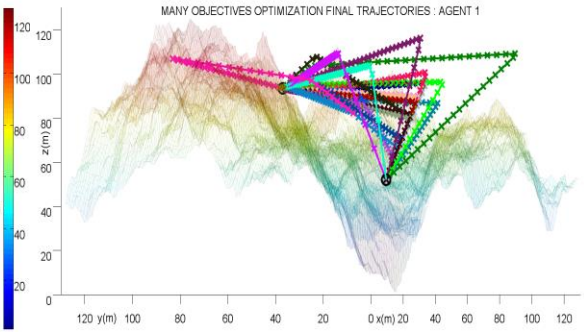
Figure 5.22(g) shows that in combination 17, all four agents fly at a close range to one another. This maintains full network connectivity between all agents. Option 19 has the lowest value for the jerk cost. On the other hand, choice 21 has the highest value for jerk cost within the final population. Figure 5.22(i) and Figure 5.22(a) show that both path combinations contain sharp turns. The difference between the two options is that the agents are able to fly across the sharp turn smoother in combination 19. Another option that provides the end user with many positive trade-off values is combination 25. Here, the multi-agents are given the shortest paths with many nodes at lower parts of the terrain. It also has the minimum value for spline deviation error. Thus, it shows that these paths are easy to track by the quadrotors.

Table 5.15 shows that there are more combinations that contain possible agent-to-agent collisions within the Mountain 1. This is because the paths of all agents fly across similar regions in comparison to the other test spaces. Still, this is based on a default safety range. The number of possible collisions can be reduced if a smaller safety radius is applied. As with all the previous environments, the end user is delivered a large number of

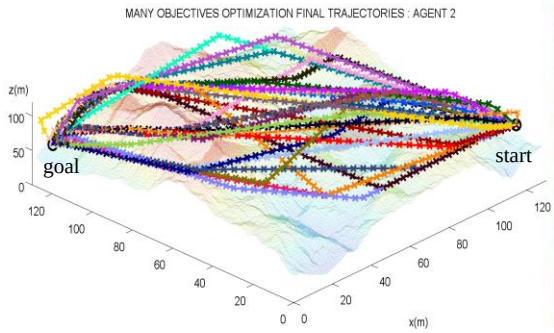
multi-agent path choices with adequate information to describe the benefits of each one. The end user is free to make a knowledgeable decision of the best trajectories for the mountainous terrain based on the trade-off values between all 30 combinations.



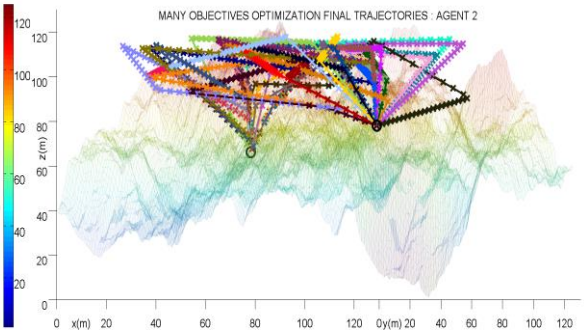
(a) Top view of agent 1's paths within the final generation.



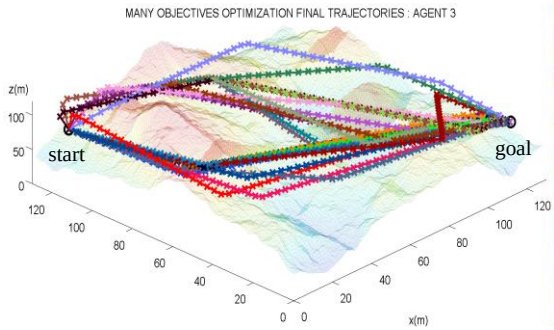
(b) Side view of agent 1's paths within the final generation.



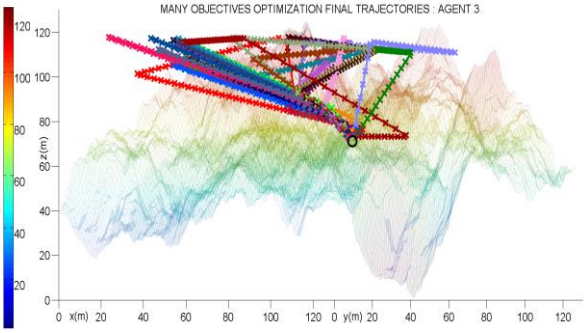
(c) Top view of agent 2's paths within the final generation.



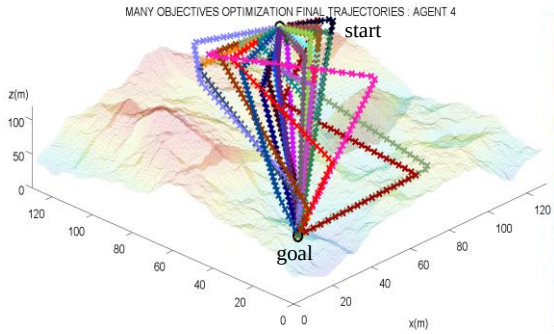
(d) Side view of agent 2's paths within the final generation.



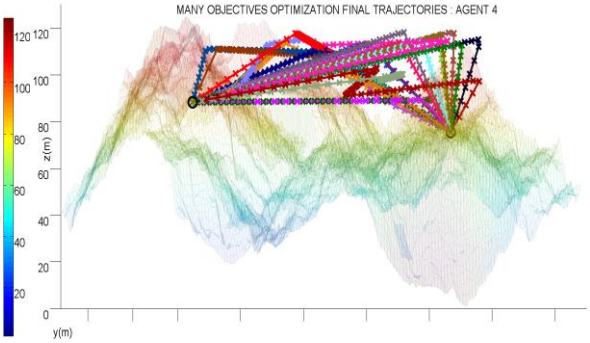
(e) Top view of agent 3's paths within the final generation.



(f) Side view of agent 3's paths within the final generation.

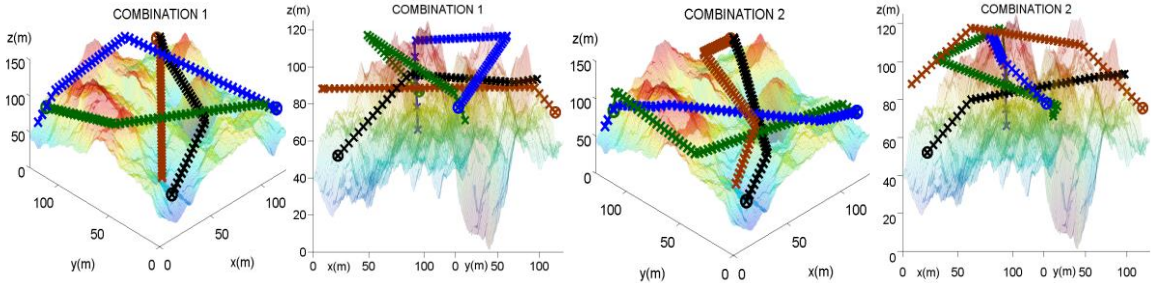


(g) Top view of agent 4's paths within the final generation.

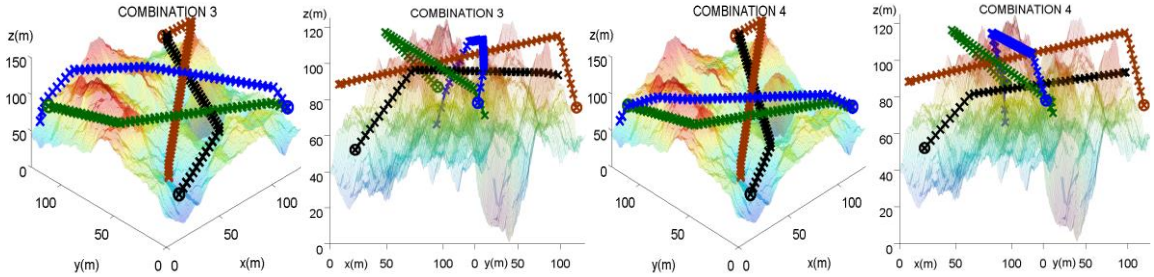


(h) Side view of agent 4's paths within the final generation.

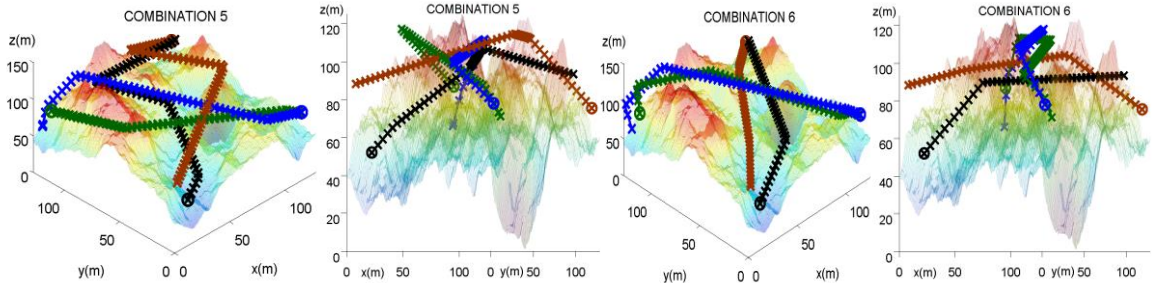
Fig. 5.20. MA-Spread Mountain 1: Final generation's multi-agent unique trajectories.



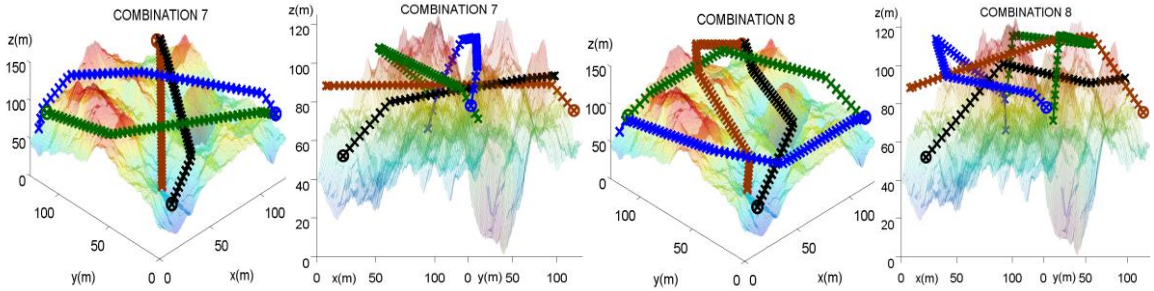
(a) Multi-agent trajectories within combination 1. (b) Multi-agent trajectories within combination 2.



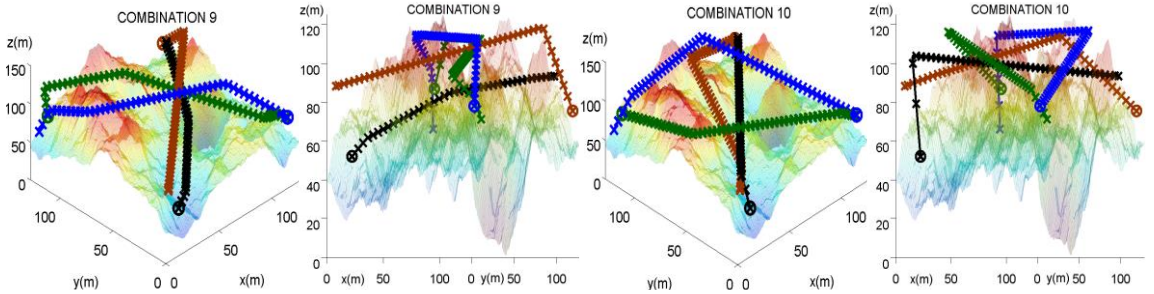
(c) Multi-agent trajectories within combination 3. (d) Multi-agent trajectories within combination 4.



(e) Multi-agent trajectories within combination 5. (f) Multi-agent trajectories within combination 6.

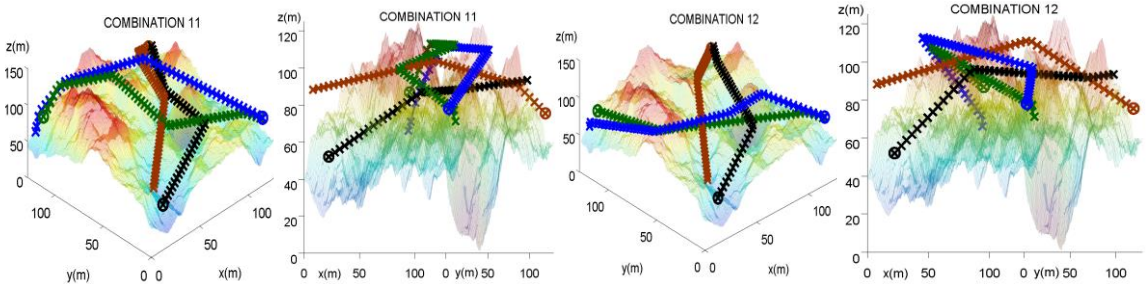


(g) Multi-agent trajectories within combination 7. (h) Multi-agent trajectories within combination 8.



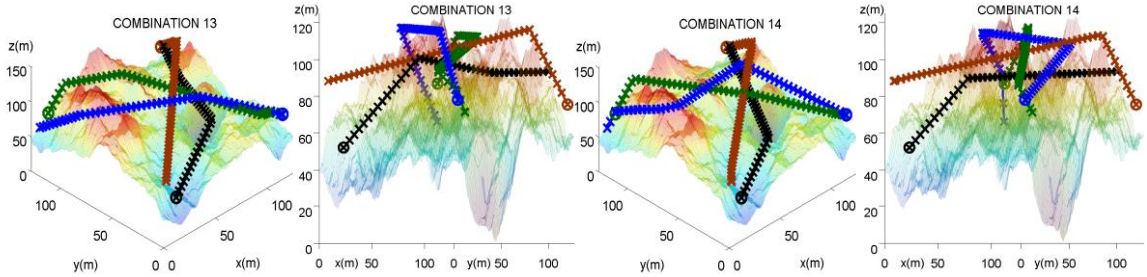
(i) Multi-agent trajectories within combination 9. (j) Multi-agent trajectories within combination 10.

Fig. 5.21. MA-Spread Mountain 1: Multi-agent path combination 1-10 for the final generation.



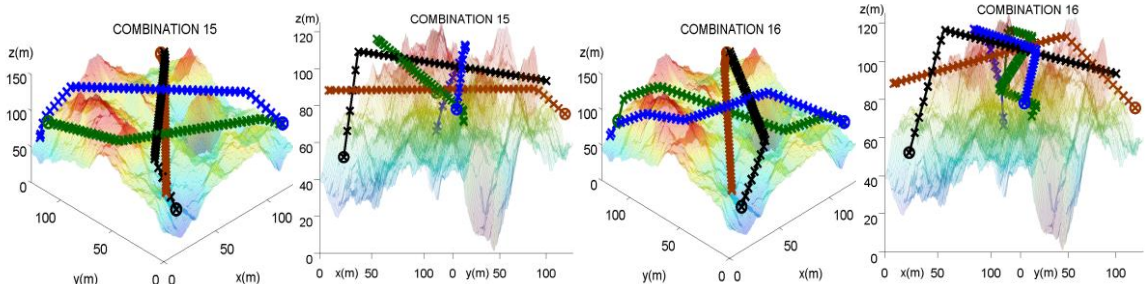
(a) Multi-agent trajectories within combination 11.

(b) Multi-agent trajectories within combination 12.



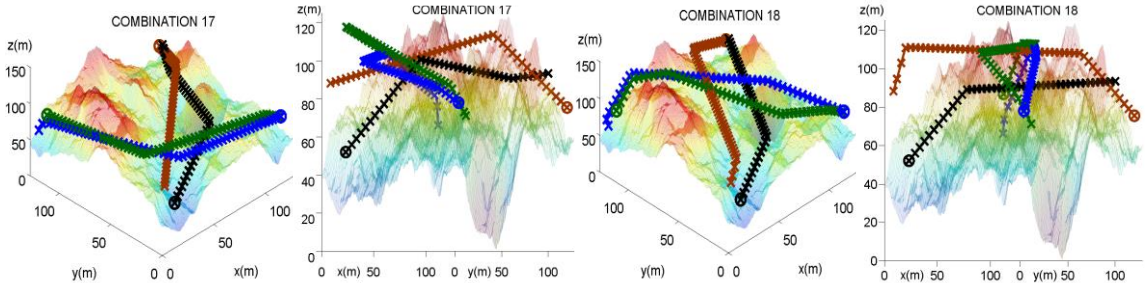
(c) Multi-agent trajectories within combination 13.

(d) Multi-agent trajectories within combination 14.



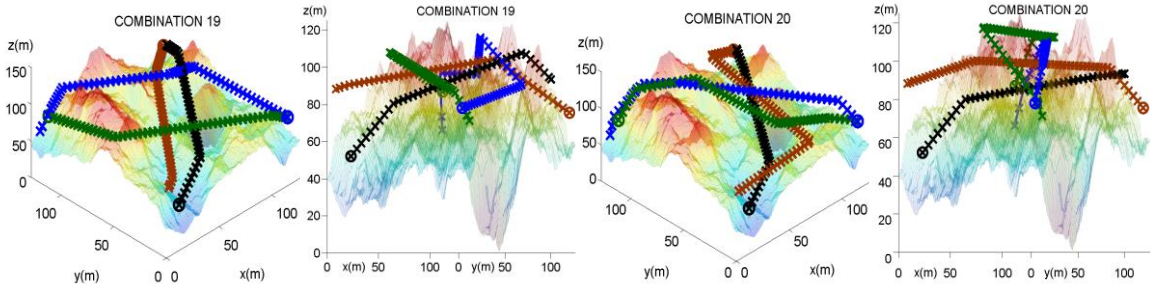
(e) Multi-agent trajectories within combination 15.

(f) Multi-agent trajectories within combination 16.



(g) Multi-agent trajectories within combination 17.

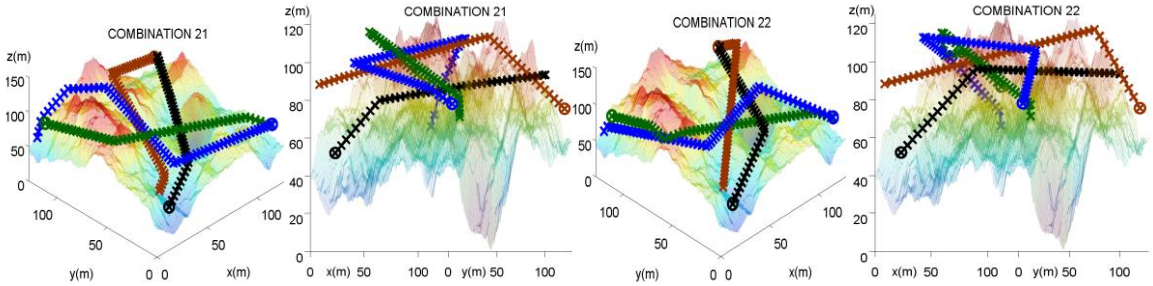
(h) Multi-agent trajectories within combination 18.



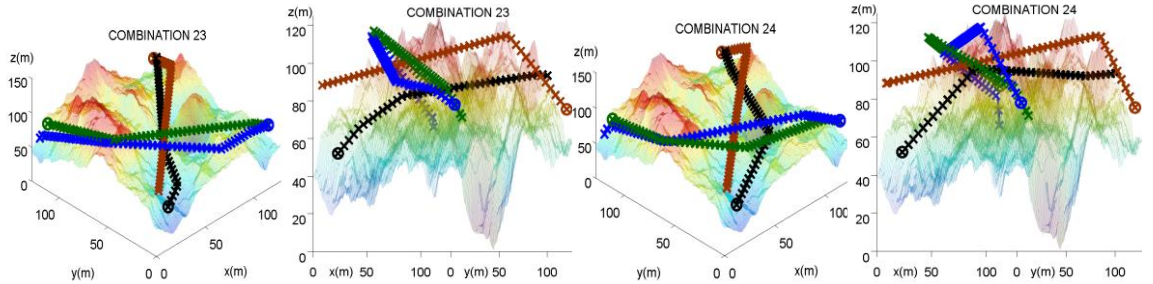
(i) Multi-agent trajectories within combination 19.

(j) Multi-agent trajectories within combination 20.

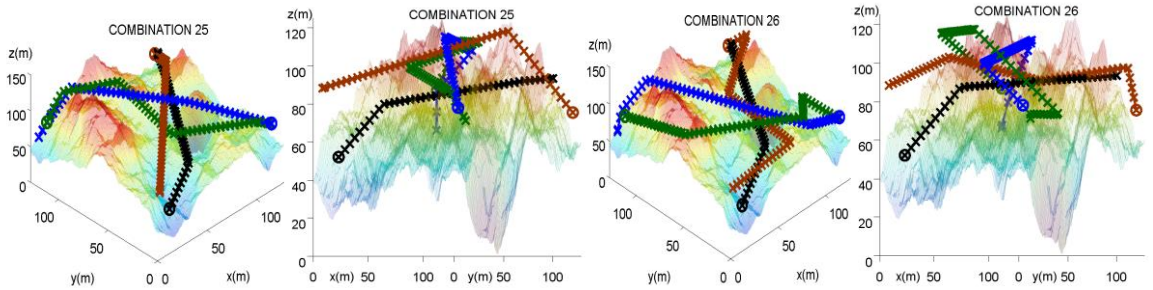
Fig. 5.22. MA-Spread Mountain 1: Multi-agent path combination 11-20 for the final generation.



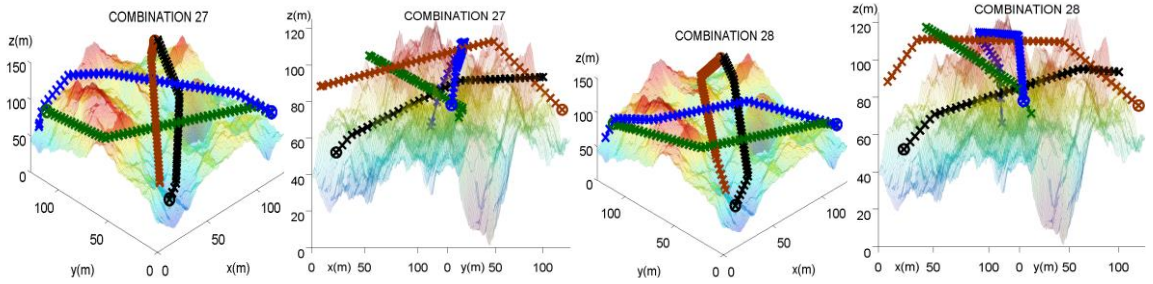
(a) Multi-agent trajectories within combination 21. (b) Multi-agent trajectories within combination 22.



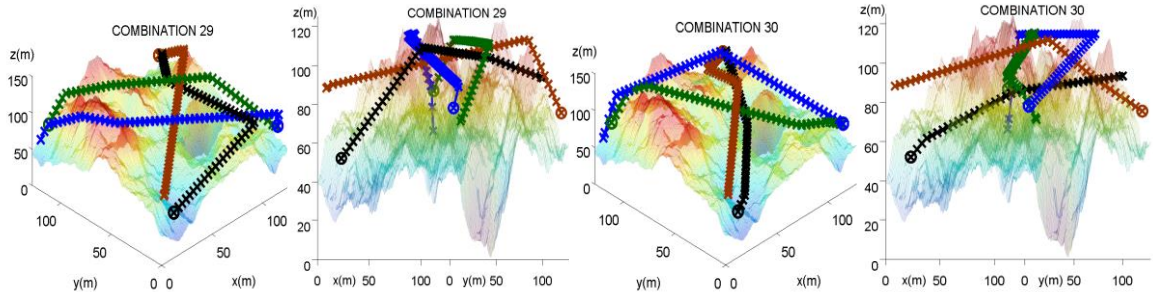
(c) Multi-agent trajectories within combination 23. (d) Multi-agent trajectories within combination 24.



(e) Multi-agent trajectories within combination 25. (f) Multi-agent trajectories within combination 26.



(g) Multi-agent trajectories within combination 27. (h) Multi-agent trajectories within combination 28.



(i) Multi-agent trajectories within combination 29. (j) Multi-agent trajectories within combination 30.

Fig. 5.23. MA-Spread Mountain 1: Multi-agent path combination 21-30 for the final generation.

TABLE 5.15: MA-Spread MOUNTAINOUS TERRAIN FINAL GENERATION'S RESULTS.

MOUNTAIN Spread COMBINATION	PATH LENGTH (*meters*)	ALTITUDE COST (*meters*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE ERROR *(m)*	TIME OPTIMALITY *(ratio)*	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE (*% free space*)	SENSORY DATA OVERLAP (*grid block overlap*)
1	775.94	97.63	4097.00	4.95	0.02	0.00	7837.64	502.00	46.00	22210.00	2533.49	18.00
2	819.03	97.26	4736.00	5.02	0.02	0.00	8388.11	678.00	0.00	31671.00	1630.52	17.00
3	827.79	101.44	4251.00	5.62	0.02	1.65	8090.87	473.00	34.00	26051.00	2874.54	17.00
4	802.37	98.19	4591.00	4.84	0.02	0.00	8123.32	558.00	0.00	27796.00	2834.86	19.00
5	860.49	98.84	4457.00	5.56	0.02	0.00	8834.80	487.00	264.00	21756.00	2655.35	20.00
6	809.20	98.50	4227.00	5.15	0.02	0.00	8842.75	482.00	278.00	30449.00	4055.77	16.00
7	791.09	97.71	4079.00	5.60	0.02	1.78	8138.84	455.00	222.00	26681.00	2685.42	20.00
8	781.37	96.63	3969.00	5.42	0.02	0.00	8318.60	533.00	462.00	27512.00	3032.11	19.00
9	797.93	95.26	4227.00	6.18	0.01	2.37	9846.23	487.00	622.00	28386.00	5008.92	19.00
10	834.97	99.43	3911.00	6.01	0.02	0.00	8190.06	416.00	0.00	24761.00	1169.71	12.00
11	854.99	97.95	4881.00	5.74	0.02	6.66	8628.93	654.00	0.00	29430.00	2698.25	21.00
12	811.36	96.26	4561.00	5.73	0.01	1.94	10115.02	528.00	278.00	23470.00	7356.25	19.00
13	783.72	96.93	4172.00	4.90	0.02	0.00	8028.42	480.00	0.00	31138.00	2089.35	19.00
14	833.09	94.96	4724.00	5.54	0.01	1.62	10260.27	483.00	0.00	34395.00	4695.93	17.00
15	785.81	97.87	3989.00	5.69	0.02	0.04	8212.90	429.00	0.00	25294.00	2181.49	17.00
16	793.98	98.40	3992.00	5.93	0.02	0.00	8171.43	483.00	0.00	30473.00	1941.90	19.00
17	769.53	95.58	3913.00	5.48	0.02	2.40	8438.44	399.00	320.00	29903.00	2166.27	20.00
18	774.76	95.54	4029.00	5.39	0.02	0.00	8858.86	533.00	432.00	28775.00	3149.36	19.00
19	849.40	98.60	4720.00	6.88	0.02	0.00	9186.07	522.00	234.00	25863.00	4267.29	19.00
20	817.60	99.15	4187.00	6.50	0.02	0.16	8993.65	422.00	556.00	27923.00	4079.19	18.00
21	775.68	97.58	4079.00	4.66	0.02	0.00	7910.34	424.00	110.00	27073.00	1943.75	16.00
22	799.89	97.68	4506.00	4.68	0.02	0.00	8099.52	504.00	82.00	27802.00	2216.82	15.00
23	790.54	96.44	4146.00	5.60	0.01	2.01	9626.14	448.00	352.00	23124.00	6853.11	18.00
24	792.75	98.09	4233.00	5.04	0.02	0.00	8758.21	469.00	278.00	27758.00	3579.86	12.00
25	769.36	93.02	4571.00	6.21	0.01	1.89	11176.87	519.00	0.00	38357.00	6644.91	15.00
26	783.02	93.10	4167.00	5.98	0.01	2.13	10421.76	583.00	292.00	43183.00	4577.02	19.00
27	810.97	93.48	4377.00	5.83	0.02	0.07	9334.68	587.00	198.00	33249.00	3010.23	20.00
28	812.59	95.20	4563.00	6.69	0.01	2.50	11464.58	405.00	328.00	33934.00	4921.29	16.00
29	777.66	93.90	4380.00	5.24	0.01	0.00	9639.19	479.00	430.00	36343.00	4853.62	20.00
30	812.67	95.59	4889.00	5.77	0.02	1.62	9481.31	495.00	252.00	40634.00	2677.00	17.00

5.4.3 TYPES OF SIMULATION MODELS

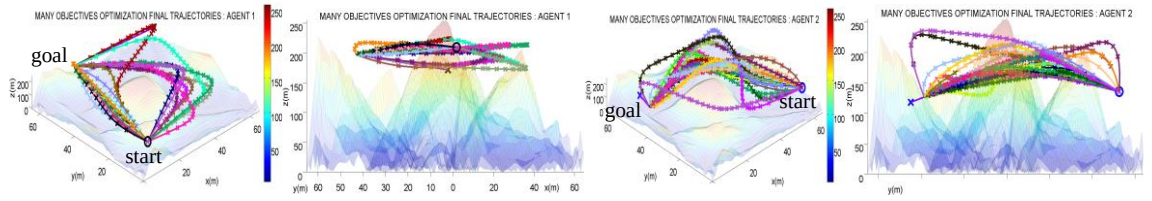
Here, three different simulation models are presented for the mountainous terrain. The first mountainous terrain was discussed in Section 5.4.1-5.4.2. This section compares the performance of the algorithm across two more terrains. These environments vary in number of randomly placed high peaks. Each terrain also varies in size. Table 5.16 shows the differences between these models. The algorithm is sped up by increasing the maximum distances between two neighbouring path nodes within Terrain 2 and 3. Only nodes that are more than 20 meters apart are padded with additional nodes between them. Similarly, a path in Terrain 2 and 3 spaces is only sampled once to generate a smooth spline.

TABLE 5.16: MA-Spread MOUNTAINOUS TERRAIN 1-3 PARAMETERS.

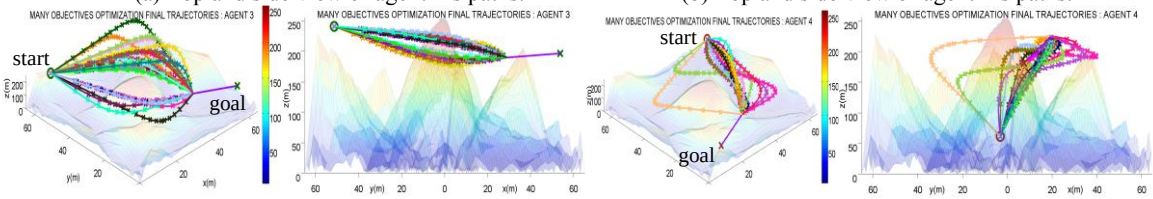
SIMULATION	MOUNTAINOUS TERRAIN 1	MOUNTAINOUS TERRAIN 2	MOUNTAINOUS TERRAIN 3
RANDOM LARGE PEAKS	15	6	6
TERRAIN SIZE	129x129x 107	65x65x 255	65x65x99m
MOUNTAIN SIZE	30m<length<80m	5m<length<48m	25m<length<40m
MAX DISTANCE BETWEEN NODES	2m	20m	20m
SPLINE SAMPLE RATE	5 samples/iteration	5 samples/simulation	5 samples/simulation

Like the previous two test spaces, the results for the Terrain 2 and 3 in Table 5.17-5.18 shows that the algorithm struggles to minimize the spline deviation error when the distances between two nodes are increased. One difference in this test space is that the objective value for flight height of a team of quadrotors also shows a slight increase. Despite this, the results show that the other 10 objectives are minimized at a larger percentage for Terrain 2 and 3 when compared to Terrain 1. These results show that the algorithm can minimize the values for at least 10 objectives for different terrains with different path resolution and running time. It able to perform across smaller and larger sized terrains with high peaks. The agents can maintain the minimum neighbour-to-neighbour distance of 2 meters across all three terrains.

The running time for all three terrains are shown in Table 5.19. The results here show similarities with both the cityscape and indoor environments. The part of the algorithm that requires the most running time in the mountainous terrain is also the collision check process. The mountainous terrain is a highly challenging environment because the entire grid is filled with obstacles in the form of large and small peaks. The results in Table confirm this because more than 80% of the processing time is dedicated to avoiding these peaks. One difference between the mountains in comparison to the other test spaces is that Mountain 3 with low resolution paths required more time than the high-resolution Mountain 1 terrain. This occurs because Mountain 3 challenges the path planner in a unique way because it has a large and wide spread mountain peak in the centre of the test



(a) Top and side view of agent 1's paths. (b) Top and side view of agent 2's paths.

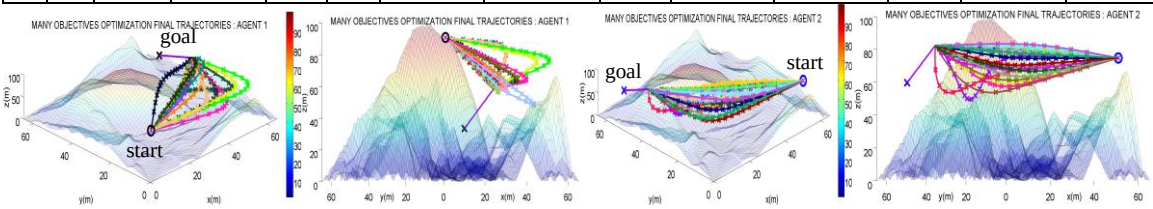


(c) Top and side view of agent 3's paths. (d) Top and side view of agent 4's paths.

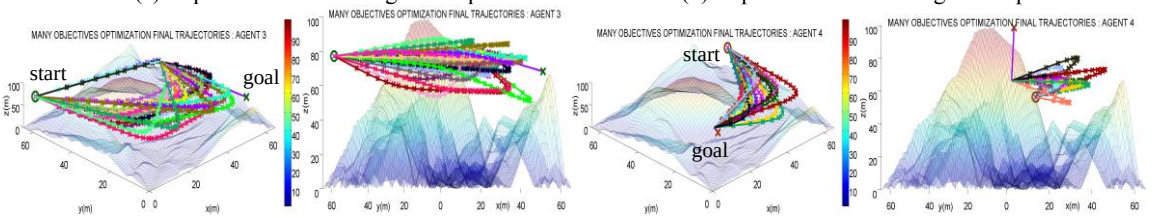
Fig. 5.24. MA-Spread Mountain 2: Final generation's multi-agent unique trajectories.

TABLE 5.17: MA-Spread MOUNT 2: MANY-OBJECTIVES OPTIMIZATION RESULTS.

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE (*no free*) *space grids*	SENSORY OVERLAP (*no grid block*) *overlap*
1	max	1709.66	212.60	564.00	47.49	0.85	5.90	1711.45	250.00	622.00	2980.00	449.26	28.00
	mean	1026.34	180.29	322.93	33.30	0.53	2.19	995.48	42.47	81.87	1129.50	108.79	16.23
	min	572.85	150.07	110.00	16.25	0.24	0.00	379.20	4.00	0.00	435.00	5.10	6.00
61	max	690.26	198.78	122.00	10.26	1.36	1.67	450.92	126.00	22.00	547.00	56.33	12.00
	mean	538.58	189.13	67.93	7.89	0.86	0.21	213.55	79.07	8.53	392.23	12.67	6.70
	min	450.22	176.03	30.00	5.63	0.57	0.00	107.27	68.00	0.00	270.00	1.33	1.00
	%	47.52	-4.90	78.96	76.31	-62.26	90.51	78.55	-86.18	89.58	65.27	88.35	58.72



(a) Top and side view of agent 1's paths. (b) Top and side view of agent 2's paths.



(c) Top and side view of agent 3's paths. (d) Top and side view of agent 4's paths.

Fig. 5.25. MA-Spread Mountain 3: Final generation's multi-agent unique trajectories.

TABLE 5.18: MA-Spread MOUNT 3: MANY-OBJECTIVES OPTIMIZATION RESULTS

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE (*no free*) *space grids*	SENSORY OVERLAP (*no grid block*) *overlap*
1	max	948.30	78.55	556.00	34.21	0.61	6.31	1396.20	137.00	582.00	2024.00	225.49	13.00
	mean	609.09	70.16	321.07	23.70	0.33	2.62	937.32	21.50	83.40	1083.43	42.25	7.40
	min	325.64	57.81	26.00	6.48	0.17	0.00	138.79	3.00	0.00	275.00	0.46	3.00
84	max	402.00	76.86	93.00	8.47	0.74	3.38	428.68	19.00	42.00	558.00	42.50	10.00
	mean	324.95	71.07	51.23	6.70	0.57	0.74	214.70	6.50	21.27	379.50	7.97	3.00
	min	272.61	64.06	18.00	4.96	0.40	0.00	115.55	4.00	4.00	240.00	0.00	0.00
	%	46.65	-1.30	84.04	71.73	-72.73	71.67	77.09	69.77	74.50	64.97	81.14	59.46

space. This reduces the amount of free space across the terrain which encourages paths that move in similar directions. The algorithm will continue to perform collision checks as it searches for diverse paths. This results in a much longer running time in comparison to Mountain 2. These results show that reducing the processing time may not be as simple as reducing the resolution of a planned path. The running time can also be influenced by the complexity of the test space as well.

TABLE 5.19: MA-Spread MOUNTAINOUS TERRAIN 1-3 RUNNING TIME.

	ALGORITHM	MOUNT 1 (sec)		MOUNT 2 (sec)		MOUNT 3 (sec)	
MA-SPREAD	GA + MA-RRF Repair + DRMOO	1664.12		644.60		3569.16	
	GA	3.80	0.23 %	1.24	0.19 %	9.26	0.26 %
	MA-RRF Repair	131.12	7.88 %	241.24	37.42 %	1427.64	40.00 %
	PD Control System	146.62	8.81 %	75.00	11.64 %	63.59	1.78 %
	Spread objectives	178.90	10.75 %	86.24	13.38 %	72.58	2.03 %
	Collision Check	1330.75	79.97 %	521.02	80.83 %	3387.32	94.91 %

5.5. MA-FORMATION DIMENSIONALITY REDUCED MANY-OBJECTIVES PARAMETERS

Firstly, the variables within the dimensionality reduced many-objectives optimization for Cityscape 1, Indoor 1 and Mountain 1 is shown in Figure 5.26 and 5.27. Table 5.20 shows the constants within the MA-Formation application. The approximation of well minimized and maintenance of diverse solutions is achieved through the combination of dimensionality reduced ranking as well as adaptive niching. Each objective subset is run during for 5 iterations with a singular run of the full set of objectives. These figures show the changes that occur within the current population in terms of dominance and diversity. These parameters show if the population remains at least as diverse as the initial MA-RRF population. It also describes if the dimensionality reduced many-objectives ranking process can increase the selection pressure within each generation.

Figure 5.26 presents the number of nondominated solutions that are obtained at each generation. It shows that there are reoccurring peaks across generations. These peaks represent the large amount of dominant solutions within the population when the full objective set is applied. In this case, diverse solutions are favoured over well minimized solutions. The ranking of solutions based on dominance is possible post dimensionality reduction. The lower peaks show that a smaller number of solutions are nondominated when objective subsets are applied instead. This shows that the dimensionality reduced objective subsets will increase the selection pressure by reducing the number of dominant solutions. It allows the algorithm to slowly move closer towards both local and global optima.

Figure 5.27 shows the changing values of the adaptive niche radius. The radius is capable of adapting to the level of diversity within the current population. This process

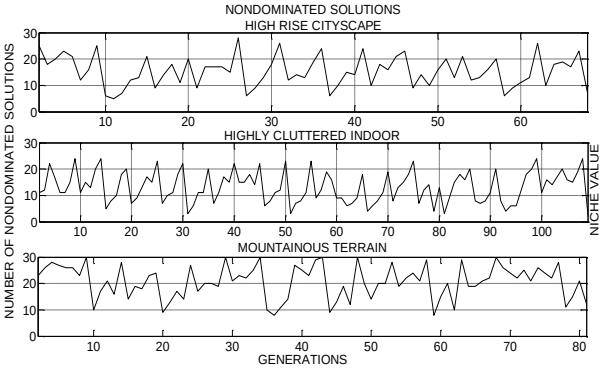


Fig. 5.26. Number of dominant solutions within the populations.

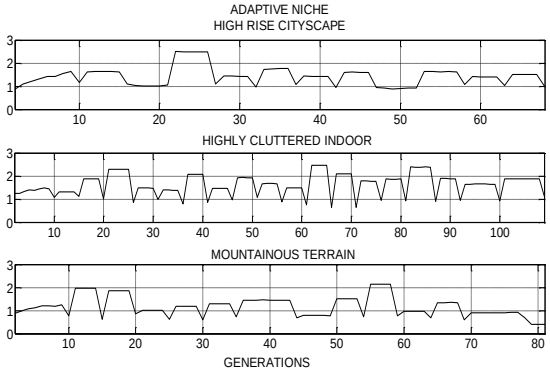


Fig. 5.27. Adaptive niche radius across generations.

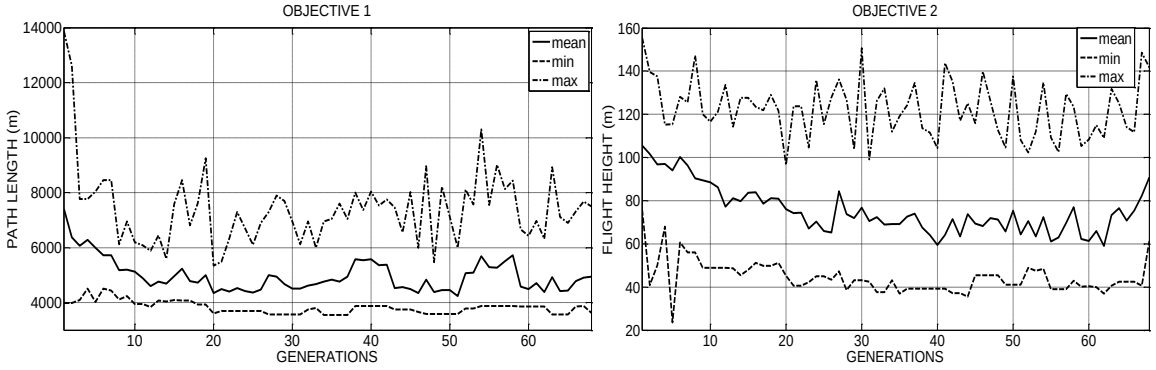
maintains representative solutions of various clusters across the generations. The average distance between the solutions and their nearest neighbours are maintained or increased throughout the generations. The only environment that shows a decline in its average distance is the mountainous terrain. In such cases, it is possible for the end user to pick a different termination point if a well spread set of the solutions is highly important.

5.6. MA-FORMATION ACROSS A HIGH-RISE CITYSCAPE

First, the algorithm is applied within the high-rise cityscape environment. There are three varieties of information that are provided to the end user. These are tables, graphs and imagery. Figure 5.28-5.29 shows the progression of the objective values at each generation for Cityscape 1. Next, Table 5.21 shows the average cost values of the entire formation reference trajectory population for the final generation. The formation reference paths for the quadrotors are presented in Figure 5.30. Lastly, Figure 5.31-5.33 shows the various formation structures that are within the final population. Table 5.22 is attached to compare of these combinations in terms of their trade off values for all objective functions. Post determination of the best combination for the end user needs can be performed based on these values. Here, we evaluate the best and worst choices between the presented formation configurations. This section shows that the hybridized algorithm successfully produces a collection of formation reference trajectories. All of the formation configurations are well optimized and have different advantages.

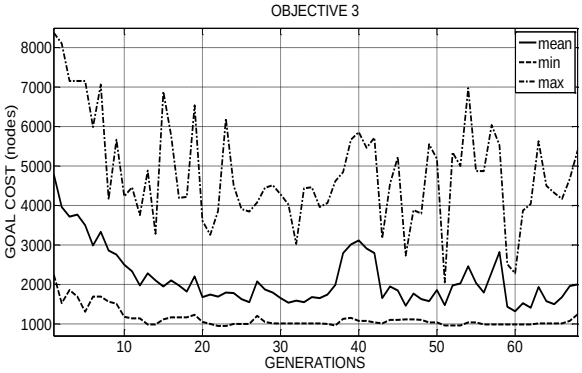
5.6.1 MANY OBJECTIVES VALUES

Similar to the MA-Spread application, the termination point is dependent on two factors. The first is the amount of cost minimization. The second criterion is the level of diversity within the population. Here, most objectives required more than 50 generations to reach a minimal value. The mean, max and min values for each objective across 68 generations are shown in Figure 5.28-5.29. The figures show that all objectives have reduced in value when compared to the initial MA-RRF path population.

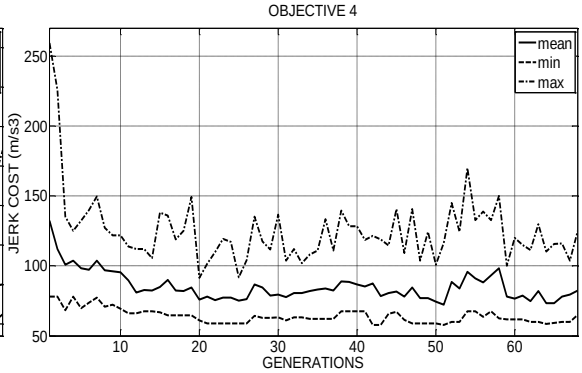


(a) Path length objective values across generations.

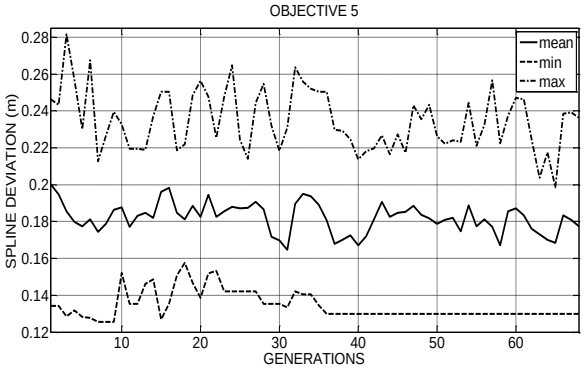
(b) Flight height objective values across generations.



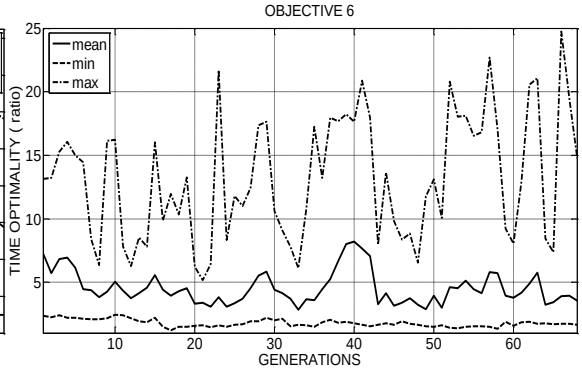
(c) Goal deviations objective values across generations.



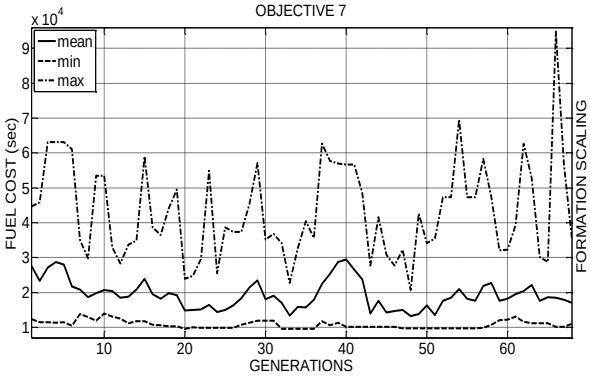
(d) Jerk cost objective values across generations.



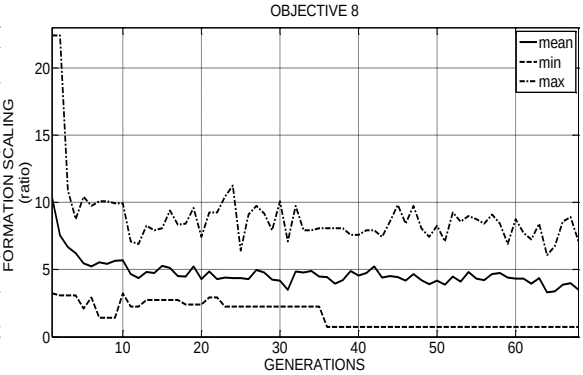
(e) Spline deviation error objective values across generations.



(f) Time optimality objective values across generations.

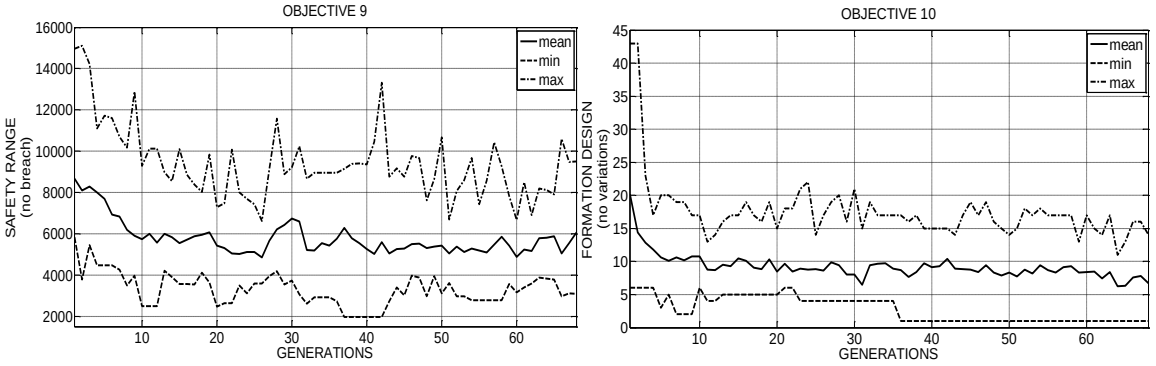


(g) Flight time objective values across generations.

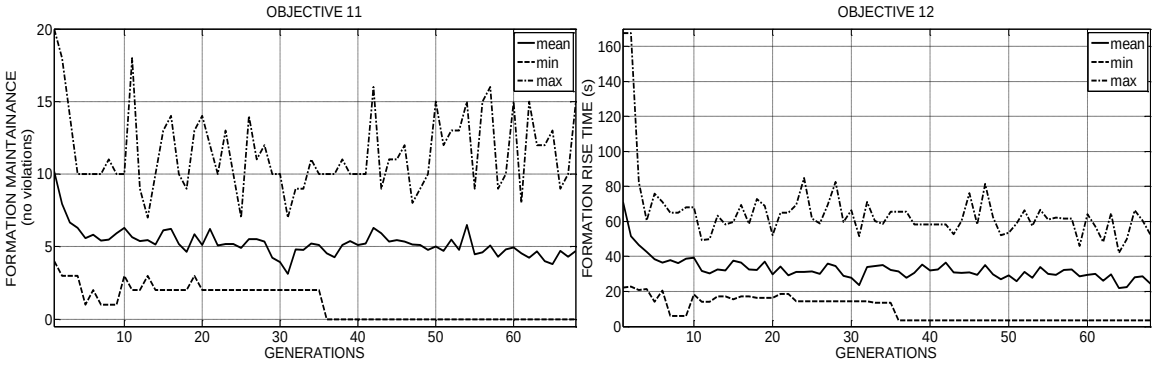


(h) Formation scaling ratio objective values across generations.

Fig. 5.28. MA-Formation Cityscape 1: Progression of the objectives 1-8 across generations.



(e) Safety zone breaches objective values across generations. (f) Formation changes objective values across generations.



(g) Formation maintenance objective values across generations. (h) Formation rise time objective values across generations.

Fig. 5.29. MA-Formation Cityscape 1: Progression of the objectives 9-12 across generations.

TABLE 5.20: MA-FORMATION PARAMETERS

Description	Value	Description	Value	Description	Value	Description	Value
Population size	30	Fyetech Agents Maximum Fuel	10*mins*	g	$9.81 ms^{-2}$	I_{YY}	$0.0086\ kgm^2$
Selection Rate	0.5	Similarity Threshold	32.5%	b	3.13×10^{-5}	I_{ZZ}	$0.0172 kgm^2$
Safety Zone Obstacles Boundary	$\xi_{obs} \pm 6$m	MA-RRF Max Repair Iterations	100	d	9×10^{-7}	c	1
Number of Gaui 330X-S Agents	4	$R_{samples}$[city, indoor, mount]	[5,2.5,4] m	m	0.4794kg	$\kappa_{threshold}$	0.8
Gaui 330X-S Agents Maximum Fuel	15*mins*	Default Number of Agents/Row	4	l	0.225m	d_{form}	2m
Number of Fyetech Agents	4	Default Number of Columns	2	I_{XX}	$0.0086\ kgm^2$	$[\ k_p\ ,\ k_d\ ]$	[0.14,0.08]

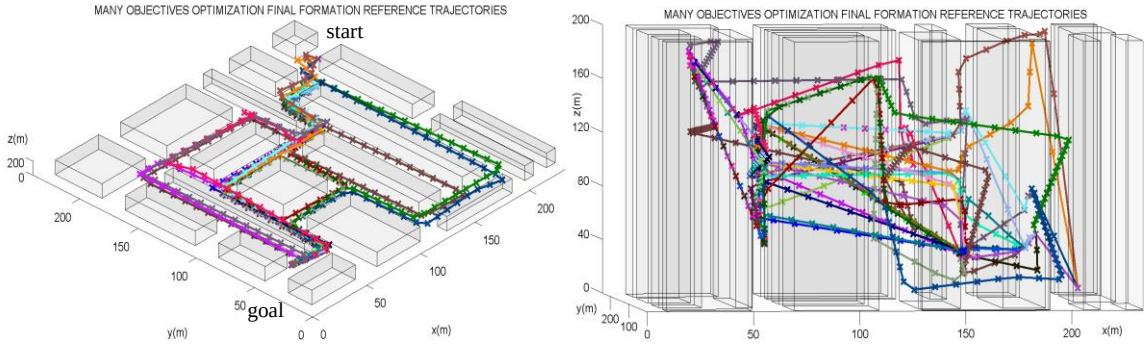
Table 5.21 shows that all the 12 objective functions have been well minimized. Two objectives were minimized with less than 30% reduction in value. The altitude cost is lessened by 14% whereas the spline deviation cost is lowered by 10%. All the other objectives are minimized by a much larger percentage. The collective path length is reduced by 33%. This means that the final generation's multi-agent formation trajectories are shorter than the initial MA-RRF paths. The number of goal node deviations and flight time is also lessened by 38%. Similarly, the quadrotors can fly at a speed that produces optimal flight time. This leads to a reduction of 51% for the time optimality cost. All of the formation trajectories are able to avoid obstacles but there are still some safety zone breaches. The amount of safety zone breaches is lowered by 30%. The formation scaling cost and rise time is lessened by 66%. The number of formation shape changes is also reduced by 67%. Lastly, the number of formation shape violations is reduced by 54%.

TABLE 5.21: MA-Formation HIGH RISE CITYSCAPE 1 EXPERIMENTAL RESULTS.

ITER		PATH LENGTH (*meters*)	ALTITUDE COST (*meters*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	SAFETY RANGE (*no breach*)	Formation SCALING *(ratio)*	Formation DESIGN (*no variations*)	Formation MAINTAINANCE (*no violations*)	Formation RISE TIME (*sec*)
1	max	13872.00	155.57	8383.00	259.16	0.25	13.13	44726.00	14974.00	22.42	43.00	20.00	167.56
	mean	7410.10	105.44	4789.20	132.45	0.20	7.20	27569.00	8658.50	10.36	20.13	10.17	71.14
	min	3987.20	75.93	2246.00	78.20	0.13	2.34	12404.00	5825.00	3.25	6.00	4.00	22.24
68	max	7504.40	141.67	5397.00	125.17	0.24	14.76	35182.00	9503.00	7.08	14.00	15.00	52.40
	mean	4942.30	90.78	1997.20	82.21	0.18	3.56	16956.00	6067.90	3.49	6.70	4.70	24.36
	min	3621.60	61.40	1249.00	65.01	0.13	1.64	11091.00	3097.00	0.75	1.00	0.00	3.51
	%	33.30	13.90	58.30	37.93	10.00	50.56	38.50	29.92	66.31	66.72	53.79	65.76

5.6.2 TRAJECTORY POPULATION

The diverse set of formation reference trajectories within the final generation of Cityscape 1 is shown in Figure 5.30. It shows that the paths are well spread across the environment and offer then end user a variety of options. It also has path nodes at various heights across the cityscape space. These formation reference trajectories will be sued to generate the formation shapes. These formation structures will then generate independent paths for all agents. Figure 5.31-5.33 shows the various trajectories for 8 quadrotors that are flying in formation.



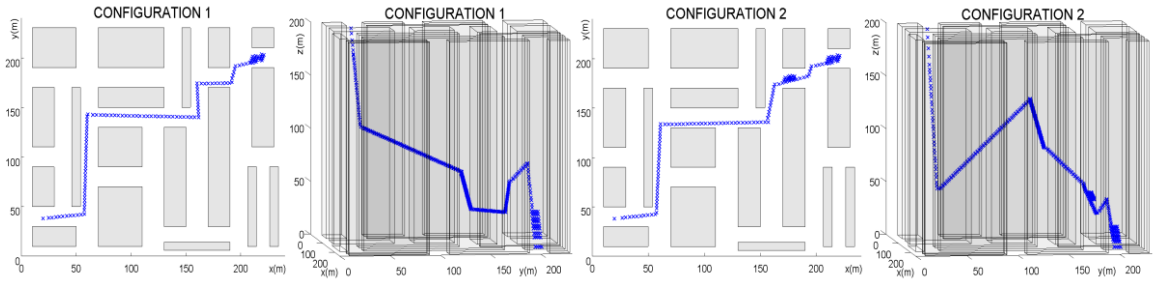
(a) Top view of formation reference paths within final generation. (b) Side view of formation reference paths within final generation.

Fig. 5.30. MA-Formation Cityscape 1: Final generation's multi-agent unique formation reference trajectories.

First, the algorithm is applied within the Cityscape 1. The results show that formation path 10 as defined in 5.31(j) produces the shortest paths for all 8 agents. This is proven in its image as well. The quadrotors will fly across the most direct route towards their goal node. Next, configuration 28 has the minimal value for the height cost. Its image in 5.31(h) shows that this path has a majority of nodes across the lower parts of the terrain.

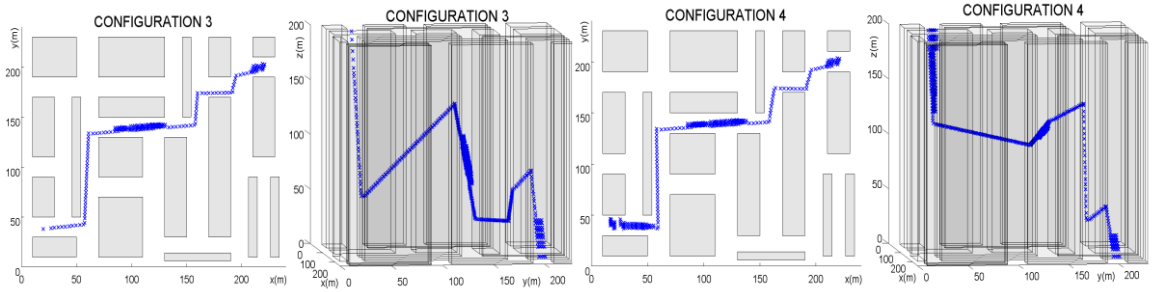
The 6th formation configuration has the lowest jerk cost and the least number of goal node deviations. 5.31(f) displays a path that isn't the most direct route but it often leans towards the destination point as it moves from node to node. The image also shows that the agents do not perform extremely sharp turns whilst flying across sharp building corners. Then, the data confirms that configuration 1 is the path that offers the least spline deviation error. It also has the minimum value for the four formation-based objective functions.

Fig. 5.31(a) shows that configuration 1 is a path that is easy to track. Its image also proves that it has the least number of formation shape changes with only one transformation from the default to danger zone formation structure. The single file formation shape proves to be a simple shape to maintain since this reference path has the minimal value for the formation maintenance, scale ratio and rise time objectives.



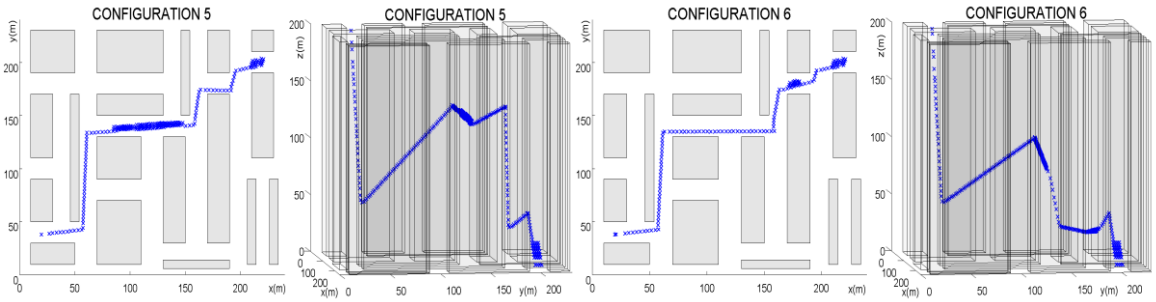
(a) Formation trajectories within configuration 1.

(b) Formation trajectories within configuration 2.



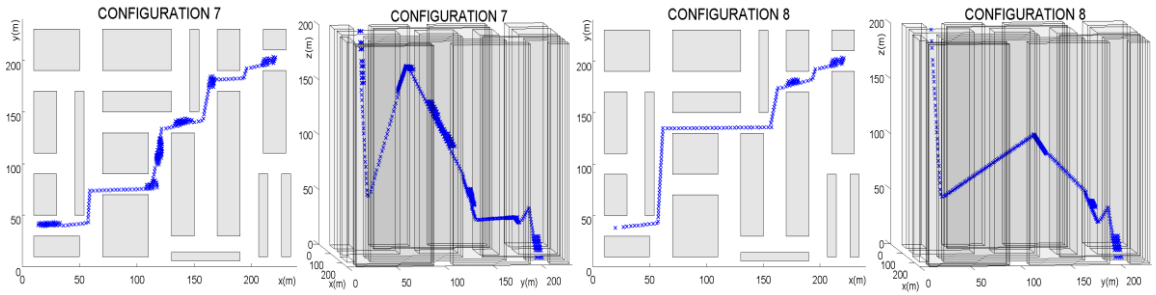
(c) Formation trajectories within configuration 3.

(d) Formation trajectories within configuration 4.



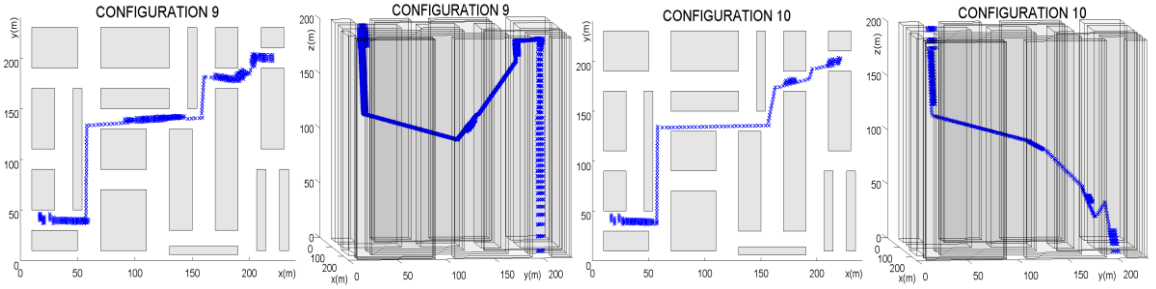
(e) Formation trajectories within configuration 5.

(f) Formation trajectories within configuration 6.



(g) Formation trajectories within configuration 7.

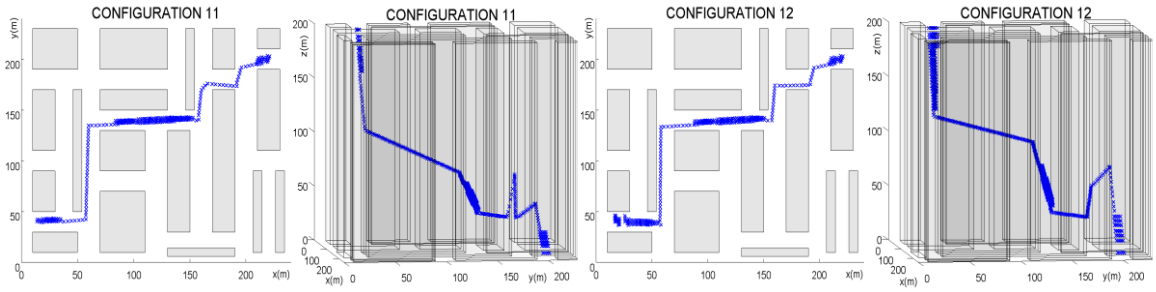
(h) Formation trajectories within configuration 8.



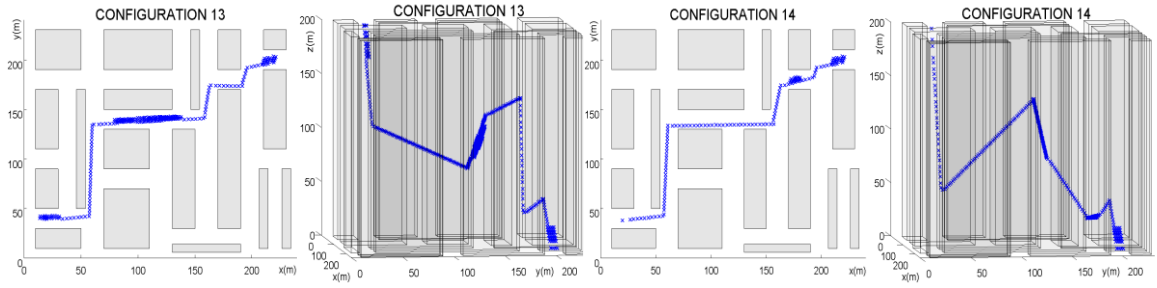
(i) Formation trajectories within configuration 9.

(j) Formation trajectories within configuration 10.

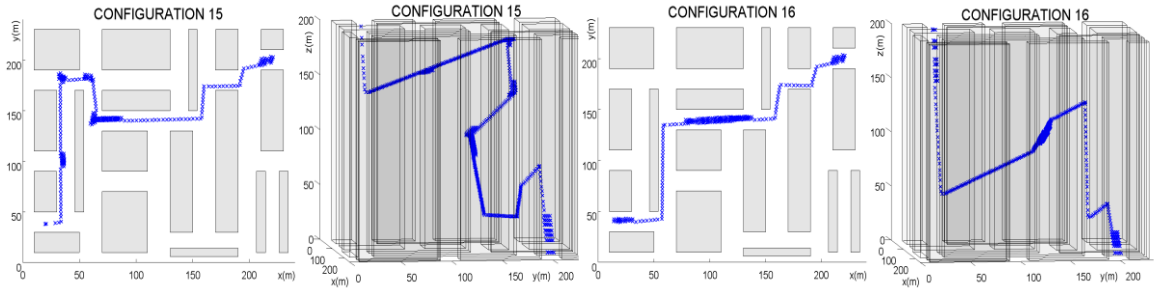
Fig. 5.31. MA-Formation Cityscape 1: Multi-agent formation configuration 1-10 for the final generation.



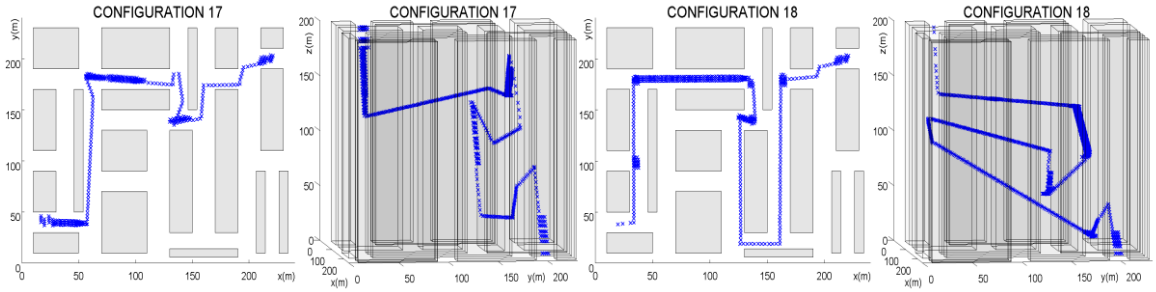
(a) Formation trajectories within configuration 11. (b) Formation trajectories within configuration 12.



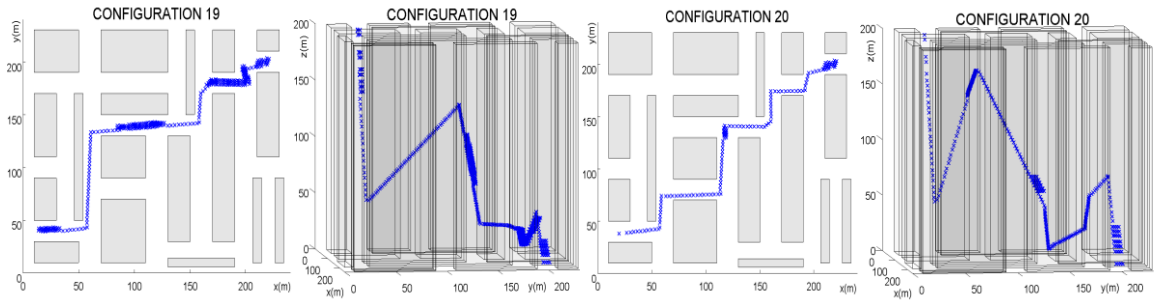
(c) Formation trajectories within configuration 13. (d) Formation trajectories within configuration 14.



(e) Formation trajectories within configuration 15. (f) Formation trajectories within configuration 16.

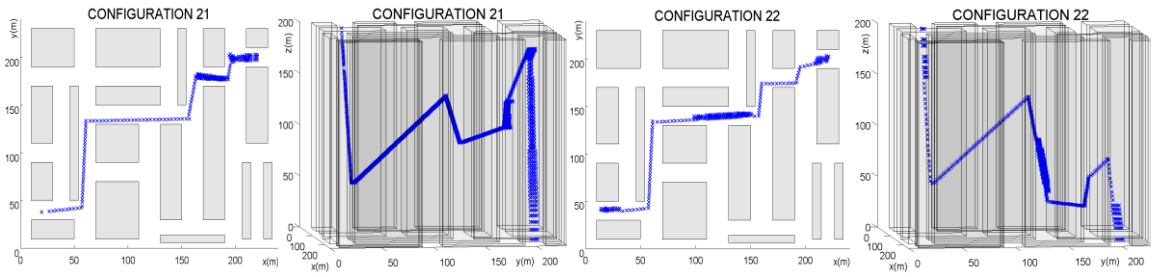


(g) Formation trajectories within configuration 17. (h) Formation trajectories within configuration 18.

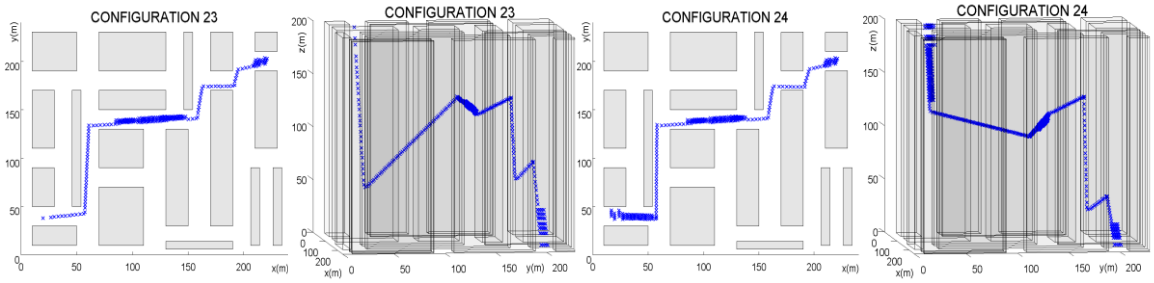


(i) Formation trajectories within configuration 19. (j) Formation trajectories within configuration 20.

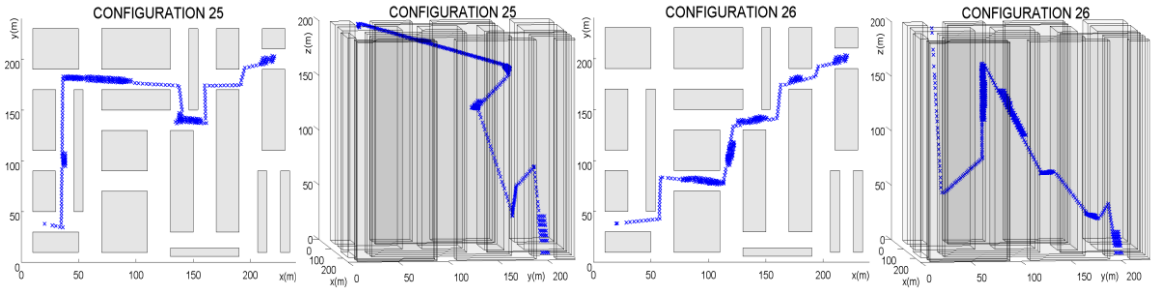
Fig. 5.32. MA-Formation Cityscape 1: Multi-agent formation configuration 11-20 for the final generation.



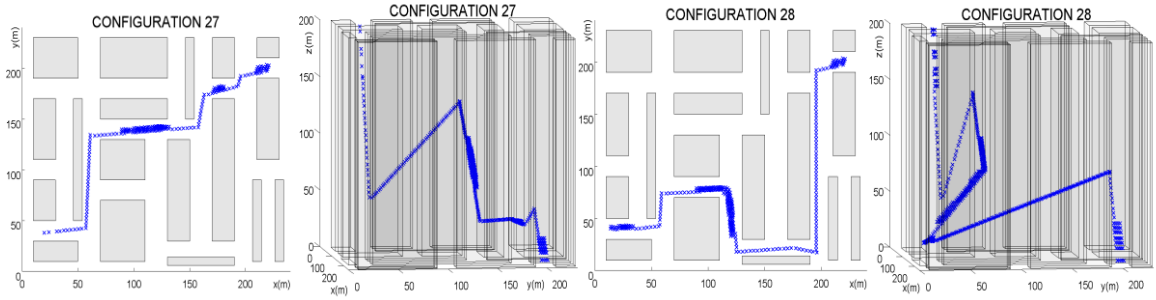
(a) Formation trajectories within configuration 21. (b) Formation trajectories within configuration 22.



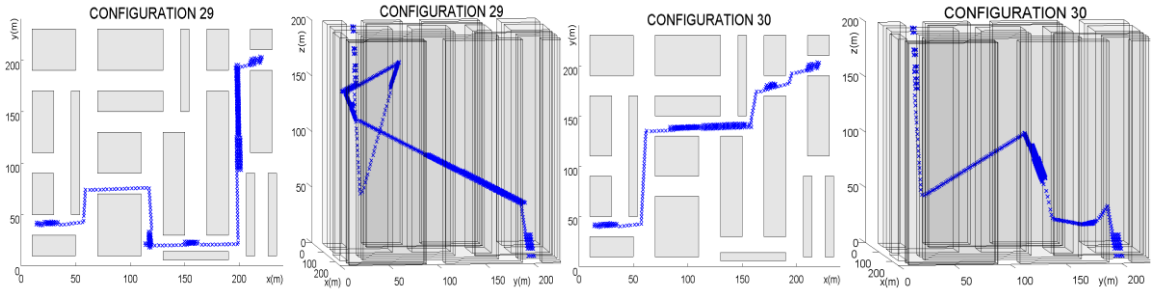
(c) Formation trajectories within configuration 23. (d) Formation trajectories within configuration 24.



(e) Formation trajectories within configuration 25. (f) Formation trajectories within configuration 26.



(g) Formation trajectories within configuration 27. (h) Formation trajectories within configuration 28.



(i) Formation trajectories within configuration 29. (j) Formation trajectories within configuration 30.

Fig. 5.33. MA-Formation Cityscape 1: Multi-agent formation configuration 21-30 for the final generation.

TABLE 5.22. MANY-OBJECTIVES VALUES OF FINAL GENERATION MA-Formation SOLUTIONS WITHIN HIGH RISE CITYSCAPE 1

FORMATION	PATH LENGTH (*meters*)	ALTITUDE COST (*meters*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio)*	FUEL COST (*sec*)	Formation SCALING (ratio)	SAFETY RANGE (*no breach*)	Formation DESIGN (*no variations*)	Formation MAINTAINANCE (*no violations*)	Formation RISE TIME (*sec*)
1	3881.78	69.11	1668.00	67.38	0.13	2.46	20619.32	0.75	4732.00	1.00	0.00	3.51
2	4222.73	87.73	1311.00	75.41	0.14	1.75	14367.82	2.08	6943.00	3.00	1.00	10.92
3	4842.24	77.44	1837.00	84.89	0.18	1.65	12362.06	2.42	5843.00	5.00	3.00	18.79
4	4508.10	103.55	1736.00	74.58	0.19	3.73	20957.01	3.25	6052.00	7.00	6.00	24.24
5	4811.37	100.96	1547.00	76.21	0.18	1.76	11842.05	2.42	5336.00	5.00	3.00	14.97
6	4133.84	72.09	1249.00	65.01	0.14	1.81	13448.12	2.08	6653.00	3.00	1.00	11.05
7	5336.90	88.91	2202.00	83.26	0.21	3.62	15060.34	7.08	5117.00	13.00	5.00	52.40
8	4029.60	79.52	1407.00	71.39	0.15	2.29	16998.73	2.08	6719.00	3.00	1.00	10.92
9	5500.35	131.01	1538.00	82.56	0.24	3.78	18775.72	4.92	5047.00	11.00	12.00	38.29
10	3621.64	90.93	1318.00	69.14	0.15	4.10	24913.41	2.75	7671.00	4.00	4.00	14.03
11	4167.28	68.17	1428.00	73.58	0.18	2.65	14320.98	2.92	6355.00	6.00	5.00	20.32
12	4315.16	80.76	1446.00	75.61	0.19	3.05	15783.24	3.25	6682.00	7.00	7.00	24.64
13	4543.93	93.67	1840.00	77.75	0.18	2.41	14519.76	2.92	5808.00	6.00	4.00	21.73
14	4317.83	82.46	1449.00	75.33	0.14	1.65	14021.98	2.08	6925.00	3.00	1.00	11.05
15	5147.14	106.61	2403.00	97.52	0.16	4.35	17313.23	5.08	7686.00	9.00	4.00	32.60
16	4885.22	87.98	1598.00	75.22	0.19	2.09	12931.85	2.92	5791.00	6.00	5.00	22.50
17	6575.93	113.59	2973.00	113.85	0.20	5.22	22374.67	4.75	6487.00	10.00	7.00	40.54
18	7504.35	90.76	5397.00	125.17	0.16	14.76	35182.02	6.42	9503.00	11.00	15.00	39.37
19	4972.19	71.16	1892.00	88.90	0.22	3.77	12806.65	4.42	4053.00	9.00	9.00	36.35
20	5449.11	85.65	2303.00	85.46	0.16	2.66	16567.31	1.75	6452.00	3.00	1.00	10.65
21	5667.64	109.09	1593.00	84.78	0.19	2.63	14932.32	2.58	6299.00	6.00	3.00	27.04
22	5048.32	77.69	1919.00	85.08	0.19	2.04	12570.86	2.92	6221.00	6.00	5.00	23.03
23	4836.96	105.47	1641.00	77.15	0.18	1.87	11978.54	2.42	5384.00	5.00	3.00	15.08
24	4312.87	103.84	1672.00	74.31	0.18	3.02	15336.59	3.08	6050.00	6.00	6.00	18.92
25	5354.37	141.67	2520.00	99.53	0.18	6.50	23124.92	4.92	7086.00	10.00	7.00	31.48
26	5269.13	84.38	2176.00	93.10	0.23	3.54	11091.38	6.92	3097.00	14.00	5.00	52.35
27	4435.37	73.80	1745.00	79.49	0.19	2.25	11581.52	3.75	4616.00	7.00	5.00	25.35
28	6272.84	61.40	3408.00	81.04	0.16	4.98	22174.22	2.92	9462.00	6.00	2.00	21.27
29	5868.15	114.18	3365.00	82.42	0.16	8.04	27069.24	4.75	3616.00	9.00	4.00	32.19
30	4438.11	69.86	1336.00	71.05	0.19	2.48	13654.08	4.08	4352.00	7.00	7.00	25.32

The data shows that choice 14 has the best value in terms of time optimality. The agents in Fig. 5.32(d) are able to fly across their trajectories at a close to ideal flight speed. Lastly, formation configuration 26 has the minimal value for flight time and number so safety zone breaches. Fig. 5.33(f) displays multi-agent paths that do not come too close to the boundaries of each building within the cityscape. This process of evaluation allows the end user to pick the best choice based on their post processing preferences.

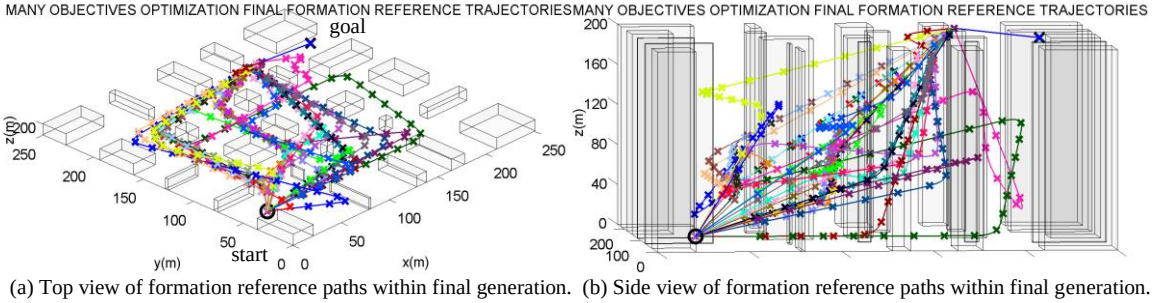
5.6.3 TYPES OF SIMULATION MODELS

This section compares three variations of the high-rise cityscape environments. The differences between each environment is similar to the MA-Spread application in Table. The results of the first cityscape environment was shown in Section 5.6.1-5.6.2. The data for Cityscape 2 and 3 is shown in Table. The simulations that were performed with Cityscape 2 and 3 prioritized lower running time in order to test the abilities of the algorithm to plan paths across a variety of test spaces.

The images show that the formation paths were planned at high resolution where the distance between two formation shapes is no more than 2 meters. This means that the agents are constantly testing the environment for nearby obstacles and the best formation shape to maintain. The downside of having a high-resolution formation planner is that it requires more running time. This is because one path can contain a large number of nodes. Likewise, the advantage of this system is its minimal spline deviation error. This is because the tracking error isn't propagated over large distances.

Figure 5.34-5.35 and Table 5.23-5.24 present the results for formation flight across Cityscape 2 and 3. The hybridized multi-agent path planner is able to generate collision free formation reference paths at the final generation for both environments. It is also able to minimize the objective values for all but the spline deviation error. The reason for this is because the node-to-node distances are much larger in order to minimize running time. This can be remedied by reducing the node-to-node distances.

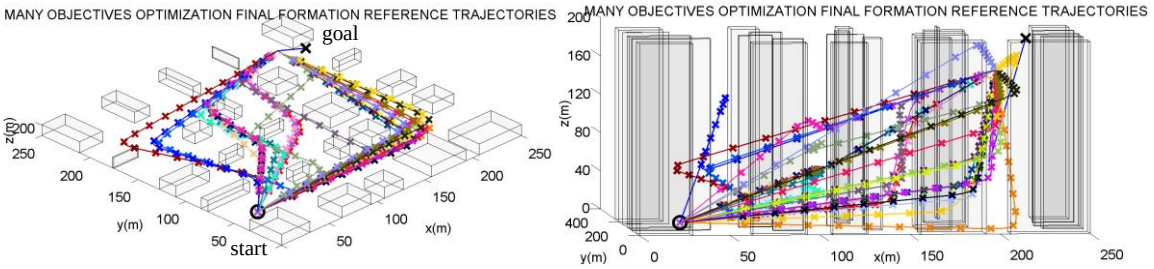
The results in Table 5.25 show the difference in running time for one generation in the three different variations of the cityscape test spaces. The data shows that there is a large difference between the results in the MA-Spread and the MA-Formation missions.



(a) Top view of formation reference paths within final generation. (b) Side view of formation reference paths within final generation.
Fig. 5.34. MA-Formation Cityscape 2: Final generation's multi-agent unique formation reference trajectories.

TABLE 5.23: MA-Formation HIGH RISE CITYSCAPE 2 EXPERIMENTAL RESULTS.

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE (*no free) space grids*	SENSORY OVERLAP (*no grid block) overlap*
1	max	13665.35	158.82	4851.00	134.13	1.24	3.76	8325.56	1025.00	27.75	78.00	34.00	571.91
	mean	6699.28	105.24	2797.17	69.86	0.69	1.60	4180.25	316.47	13.53	36.00	14.63	258.56
	min	2794.21	52.17	1005.00	23.22	0.37	0.35	1078.29	32.00	4.58	12.00	4.00	87.02
89	max	5825.25	142.51	1857.00	38.65	1.06	3.24	2631.69	793.00	11.25	28.00	15.00	211.09
	mean	3298.11	87.49	1191.70	27.47	0.81	1.29	1317.84	195.00	6.17	15.80	5.77	119.28
	min	2687.42	60.37	1007.00	19.01	0.52	0.82	1068.25	0.00	1.50	5.00	1.00	37.58
	%	50.77	16.87	57.40	60.68	-17.39	19.74	68.47	38.38	54.40	56.11	60.56	53.87



(a) Top view of formation reference paths within final generation. (b) Side view of formation reference paths within final generation.

Fig. 5.35. MA-Formation Cityscape 2: Final generation's multi-agent unique formation reference trajectories.

TABLE 5.24: MA-Formation HIGH RISE CITYSCAPE 3 EXPERIMENTAL RESULTS.

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE (*no free) space grids*	SENSORY OVERLAP (*no grid block) overlap*
1	max	13902.94	157.59	5199.00	137.65	1.04	4.29	9500.91	954.00	19.00	52.00	27.00	376.06
	mean	6640.78	108.86	2812.23	70.79	0.57	2.24	4414.45	220.90	10.75	29.03	12.83	201.42
	min	2947.16	49.95	1067.00	18.39	0.40	0.40	1175.29	0.00	1.08	3.00	1.00	23.31
62	max	3834.34	142.56	1487.00	34.08	0.95	2.10	1533.00	486.00	8.50	23.00	13.00	178.55
	mean	3050.60	73.48	1272.67	24.07	0.71	1.57	1335.17	141.87	4.54	12.00	5.13	86.06
	min	2756.27	36.26	1020.00	16.40	0.40	1.15	1068.79	0.00	0.00	1.00	0.00	0.00
	%	54.06	32.50	54.75	66.00	-24.56	29.87	69.75	35.78	57.77	58.66	60.02	57.27

Here, a much smaller percentage of the running time is spent on collision check. This is because the algorithm only generates one formation reference trajectory. Thus, it is enough to perform collision check for one agent as opposed to the entire team. Another difference between the two missions is the running time of the parallel run control system. The MA-Formation mission requires more time to simulate the movements of eight agents.

The data shows that the formation planner, uses a large percentage of the running time as well. This process includes sampling of the obstacle boundary planes and the identification of the nearest sample point for each path node. This process can be time consuming because each obstacle will contain many sampling points across it boundaries. Then, the algorithm generates the obstacle free space surface for the formation shape planning. This part is completed at a faster rate. Table shows that Cityscape 2 and 3 run the MA-Formation algorithm at a much faster rate than Cityscape 1. As with the MA-Spread simulations, the running time can be manipulated by reducing the resolution of the path planning process.

TABLE 5.25: MA-Formation HIGH RISE CITYSCAPE RUNNING TIME.

	ALGORITHM	CITYSCAPE 1 (sec)		CITYSCAPE 2 (sec)		CITYSCAPE 3 (sec)	
MA-FORMATION	GA + MA-RRF Repair + DRMOO	3641.21		641.91		562.39	
	GA	1.14	0.03 %	0.34	0.05 %	0.52	0.09 %
	MA-RRF Repair	144.82	3.98 %	62.27	9.70 %	35.67	6.34 %
	Identify nearest obstacle sample node	835.80	22.95 %	117.19	18.26 %	123.56	21.97 %
	Generate free space surface	527.60	14.49 %	109.92	17.12 %	63.21	11.24 %
	PD Control System	848.87	23.31 %	210.25	32.75 %	216.66	38.52 %
	Formation objectives	460.40	12.64 %	44.19	6.88 %	46.07	8.19 %
	Collision Check	265.84	7.30 %	103.20	16.08 %	62.93	11.19 %

5.7. MA-FORMATION ACROSS A HIGHLY CLUTTERED INDOOR ENVIRONMENT

The next simulated environment for the MA-Formation application is the highly cluttered indoor space. This environment challenges the formation planner to design shapes that continue to expand and contract across the entry points. It must be able to detect the boundaries of all clutter and design the most suitable formation structure for each path waypoint.

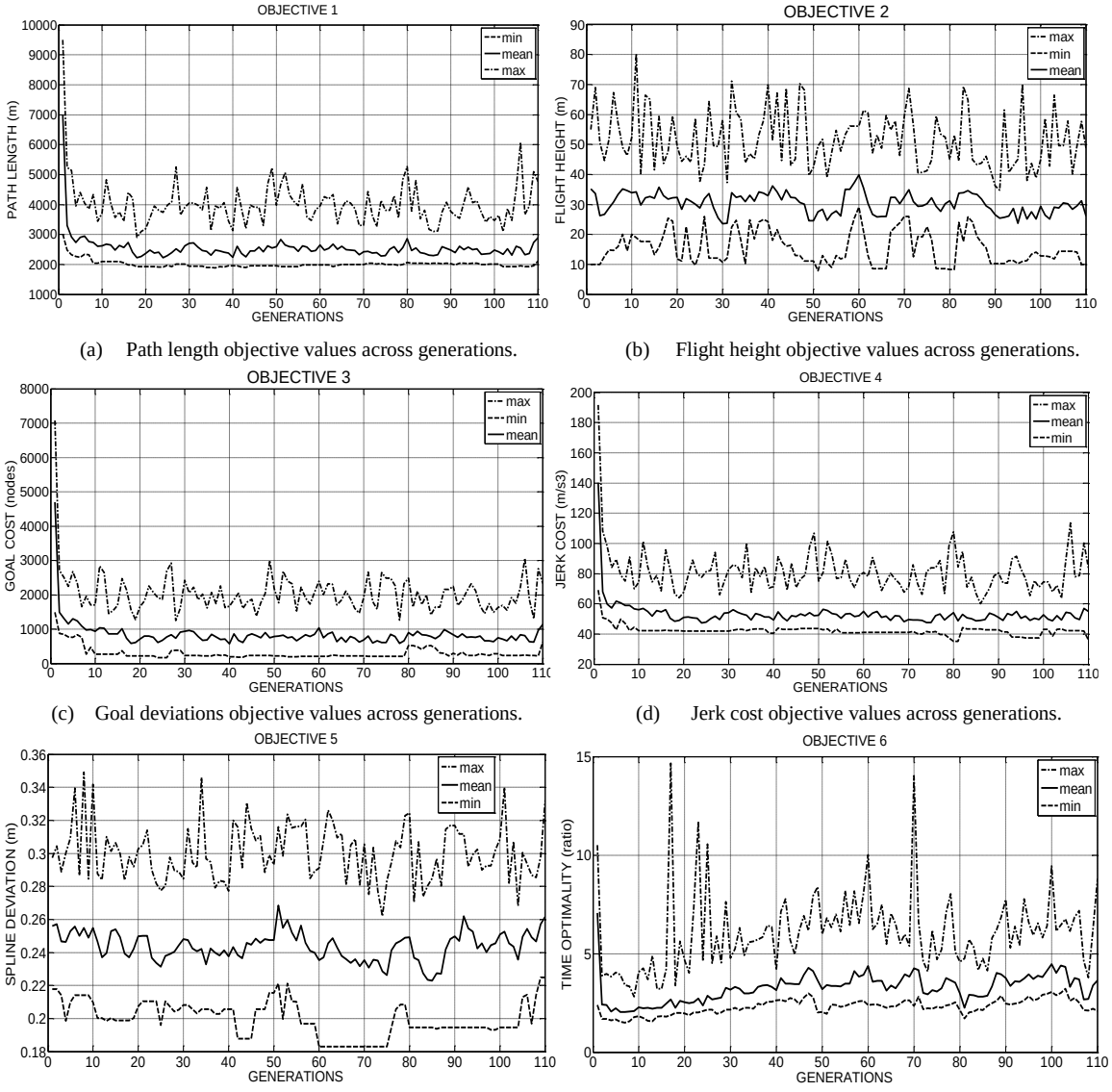
The results for the Indoor 1 test space are presented in this section. Figures 5.36 and 5.37 shows the progression of the 12 objectives across all generations for Indoor 1. Table 5.26 shows the average values of all path combinations for each objective within the final generation. The termination point for the indoor space is set at the 110^{th} generation. This is the point where most objectives have been sufficiently minimized and path diversity is maintained. Next, the unique paths for each agent are shown in Figure 5.38. These paths will form the path combinations that are shown within Figure 5.39-5.41. The cost values for all 30 formation configurations are defined in Table 5.27.

5.7.1. MANY OBJECTIVES VALUES

Next, the objective values for the formation trajectories within the Indoor 1 environment are presented in Figure 5.36-5.37. Out of all the experiments, this MA-Formation application required more than 100 iterations for the objectives to settle into their minimum values. In this case, almost all of the 12 objectives reached their minimal values after the 80^{th} generation. This is because the environment is extremely challenging for formation flight. The agents must expand and contract often as they fly across rooms. The agents must do so much more often than the cityscape or mountainous terrain. This means that the formation-based objective functions will influence the values of all of the other cost functions as well.

Similar to the cityscape environment, Table 5.26 shows that all 12 objective functions have been well minimized or maintained. The spline deviation error has been maintained across generations. The altitude cost has a 26% reduction in value when

compared to the initial MA-RRF paths. Next, the agents fly at a closer to optimal speed with a minimization of 48% in the time optimality ratio. All of the other objectives have been minimized with a more than 50% reduction in value. The multi-agent quadrotor formation path lengths are 59% shorter than the first generations. The number of goal deviations is also reduced by a large amount of 76%. Each formation trajectory is less aggressive since the jerk cost is minimized by 61%. The amount of fuel that is required by the team of agents is lessened by 63%. An important objective within the indoor space is the number of safety zone breaches. The formation planner is capable of producing trajectories that avoid all obstacles. Still, some safety zones may be breached by the formation structure. Here, the number of safety breaches is lowered by 57%. Lastly, all of the formation-based objectives are reduced in value by more than 57% each.



(a) Path length objective values across generations.

(b) Flight height objective values across generations.

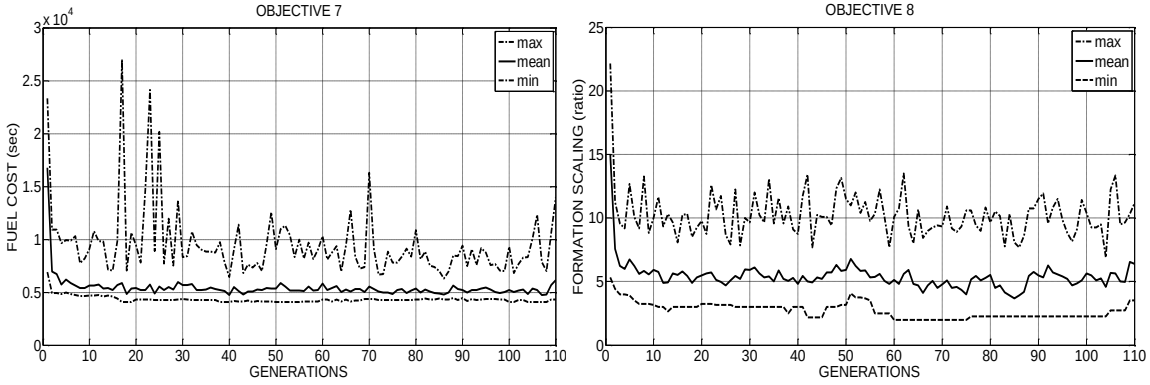
(c) Goal deviations objective values across generations.

(d) Jerk cost objective values across generations.

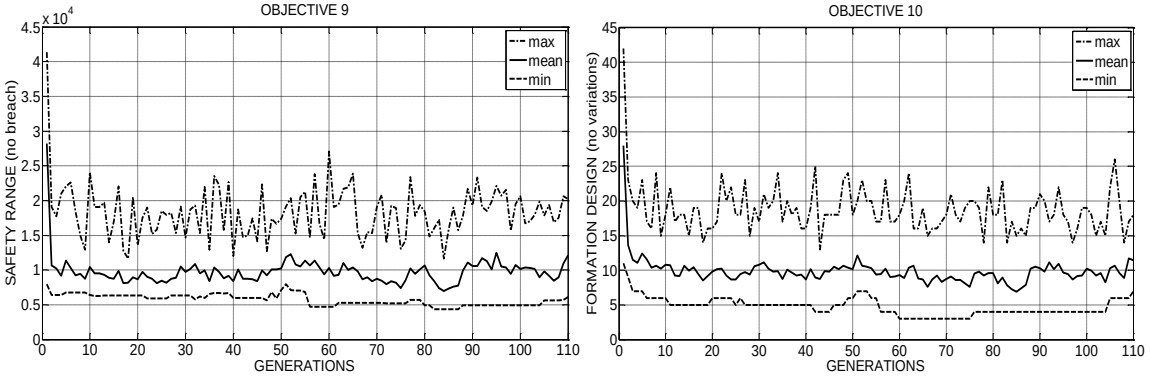
(e) Spline deviation error objective values across generations.

(f) Time optimality objective values across generations.

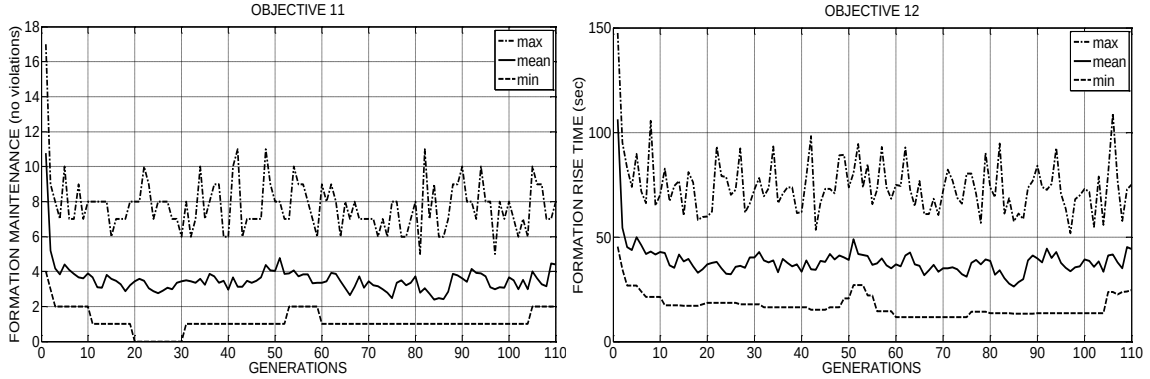
Fig. 5.36. MA-Formation Indoor 1: Progression of the objectives 1-6 across generations.



(a) Flight time objective values across generations. (b) Formation scaling ratio objective values across generations.



(c) Safety zone breaches objective values across generations. (d) Formation changes objective values across generations.



(e) Formation maintenance objective values across generations. (f) Formation rise time objective values across generations.

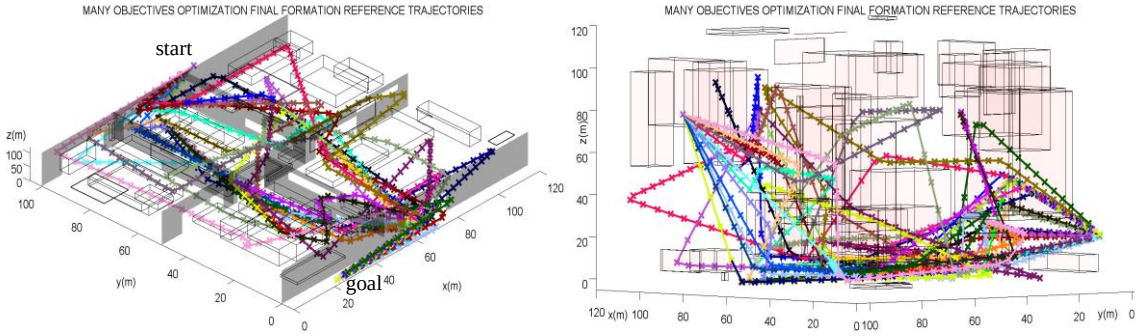
Fig. 5.37. MA-Formation Indoor 1: Progression of the objectives 6-12 across generations.

TABLE 5.26: MA-Formation HIGHLY CLUTTERED INDOOR 1 EXPERIMENTAL RESULTS

ITER		PATH LENGTH (*meters*)	ALTITUDE COST (*meters*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	SAFETY RANGE (*no breach*)	Formation SCALING *(ratio)*	Formation DESIGN (*no variations*)	Formation MAINTAINANCE (*no violations*)	Formation RISE TIME (*sec*)
1	max	9492.40	55.02	7077.00	191.65	0.30	10.51	23327.00	41386.00	22.17	42.00	17.00	147.44
	mean	6999.70	35.21	4695.10	140.10	0.26	7.06	16766.00	28167.00	15.00	28.00	10.77	106.36
	min	3022.90	9.95	1485.00	69.10	0.22	2.40	6919.40	7988.00	5.33	11.00	4.00	45.42
110	max	4765.80	48.79	2332.00	83.53	0.33	8.80	14009.00	20190.00	11.17	18.00	8.00	75.14
	mean	2885.90	26.17	1137.90	54.95	0.26	3.64	6163.70	12150.00	6.39	11.43	4.40	44.16
	min	2109.10	9.91	623.00	35.93	0.22	2.09	4303.50	6152.00	3.50	7.00	2.00	24.67
	%	58.77	25.67	75.76	60.78	0.00	48.44	63.24	56.86	57.40	59.18	59.15	58.48

5.7.2. TRAJECTORY POPULATION

Next, the unique formation reference trajectories within the final population for Indoor 1 are presented in Figure 5.38. The image shows that the path planner faced the same challenges within the MA-Spread application. Many path nodes are placed at lower parts of the terrain where there is less clutter. There are some nodes that are spread across the higher parts of each room. Still, there is diversity in terms of room exploration. It can be seen that the paths are well spread across the different rooms and entry points.

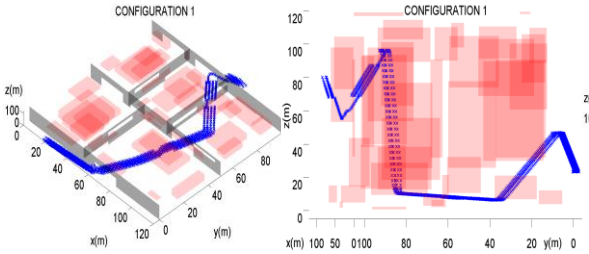


(a) Top view of formation reference paths within final generation. (b) Side view of formation reference paths within final generation.

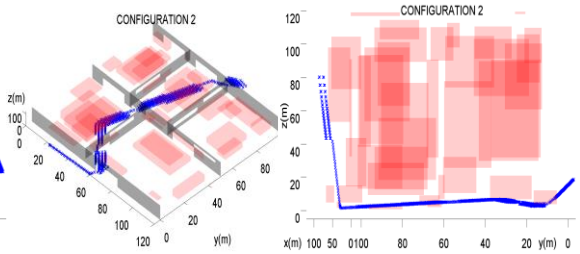
Fig. 5.38. MA-Formation Indoor 1: Final generation's multi-agent unique formation reference trajectories.

The independent trajectories for each quadrotor are shown in Figures 5.39-5.41. The formation configuration that produces the shortest paths is option 28. This choice also has the minimal value for both the jerk cost and spline deviation error. In Fig.5.41(h), it can be seen that this path is the most direct route out of all the other options. The agents will be able to fly beneath the clutter easily and move straight towards the goal node without many sharp turns. Next, formation configuration 11 has the lowest value for the altitude cost. Fig.5.40(a) clearly shows that the path nodes are mostly spread across the lower ends of the rooms. Choosing option 19 means that the end user obtains multi-agent formation paths that have the minimum value for the node-to-node goal deviation objective. Fig.5.40(i) shows that this option doesn't have the most direct paths but it keeps moving towards the goal node as it progresses from node-to-node.

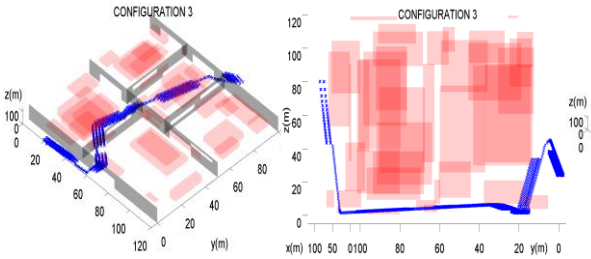
Fig.5.39(b) displays configuration 2 with formation paths that moves swiftly towards the destination point. Here, the quadrotors will be capable of flying at a close to ideal speed and reach within the minimum flight time. Option 4 has multiple advantages such as having paths that produce the minimal values for safety zone breaches, number of formation shape changes and scaling ratio. It can be seen in Fig.5.39(d) that this choice has the least number of formation structure changes when compared to the others. Lastly, option 1 has the lowest cost values for the formation maintenance and rise time objectives. This means that the agents will be capable of maintaining the adaptive formation shapes that were designed by the high-resolution formation shape planner.



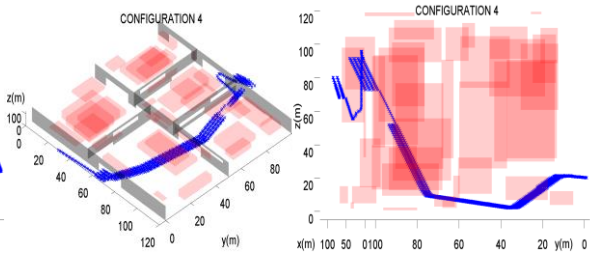
(a) Formation trajectories within configuration 1.



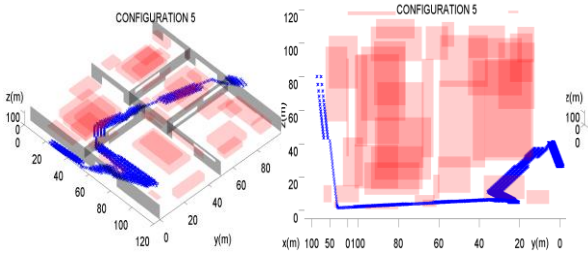
(b) Formation trajectories within configuration 2.



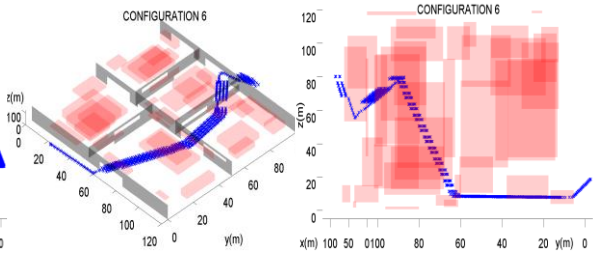
(c) Formation trajectories within configuration 3.



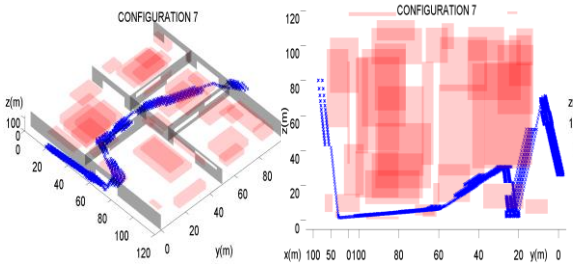
(d) Formation trajectories within configuration 4.



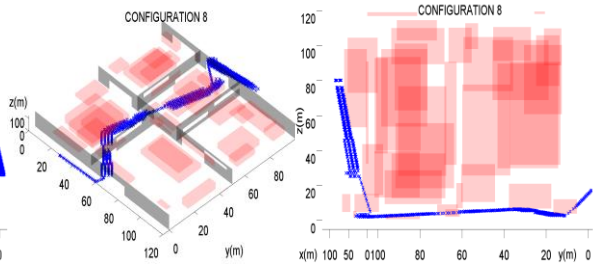
(e) Formation trajectories within configuration 5.



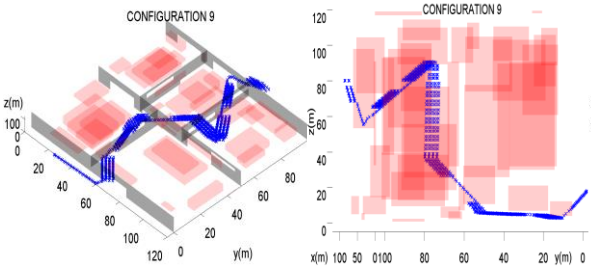
(f) Formation trajectories within configuration 6.



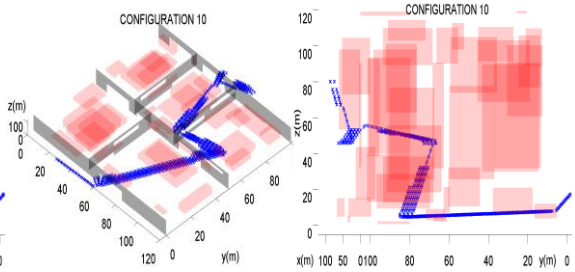
(g) Formation trajectories within configuration 7.



(h) Formation trajectories within configuration 8.

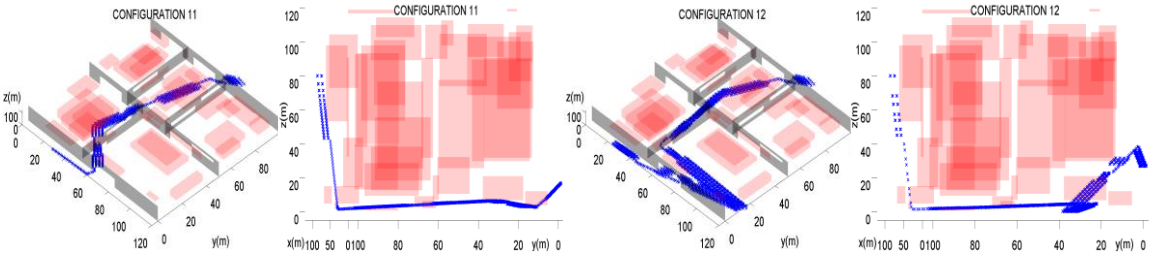


(i) Formation trajectories within configuration 9.



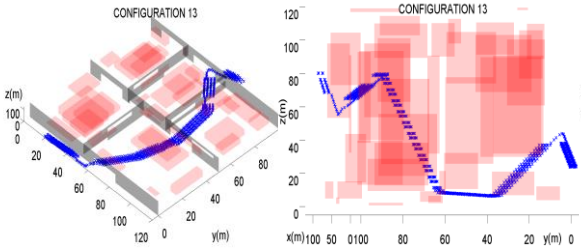
(j) Formation trajectories within configuration 10.

Fig. 5.39. MA-Formation Indoor 1: Multi-agent formation configuration 1-10 for the final generation.

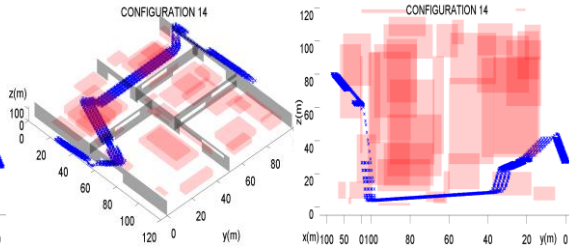


(a) Formation trajectories within configuration 11.

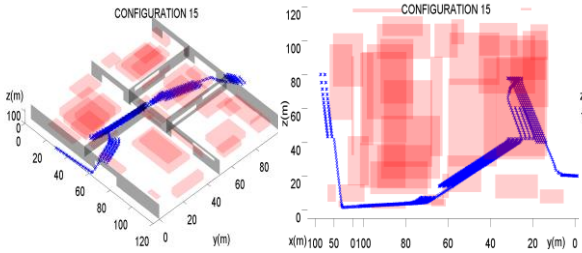
(b) Formation trajectories within configuration 12.



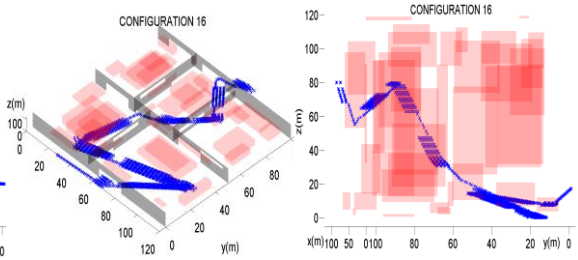
(c) Formation trajectories within configuration 13.



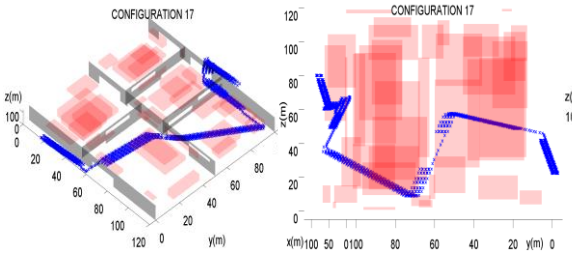
(d) Formation trajectories within configuration 14.



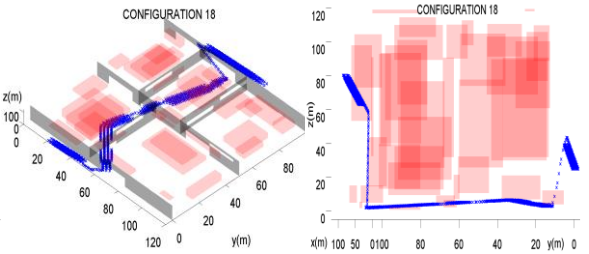
(e) Formation trajectories within configuration 15.



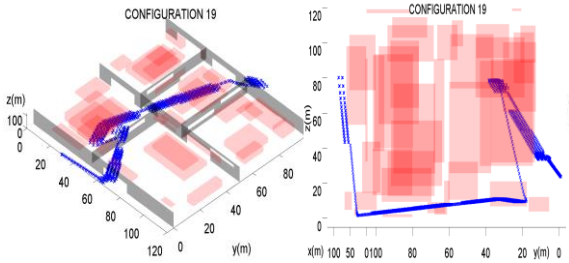
(f) Formation trajectories within configuration 16.



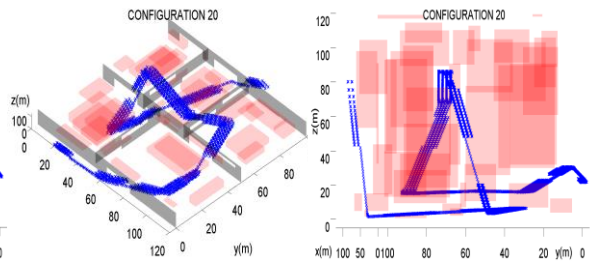
(g) Formation trajectories within configuration 17.



(h) Formation trajectories within configuration 18.

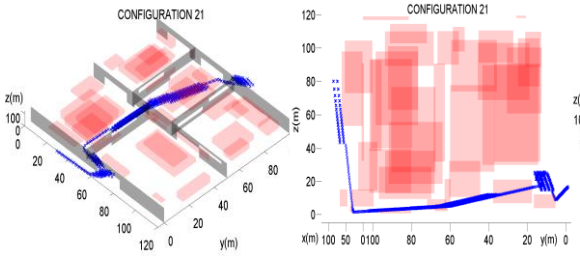


(i) Formation trajectories within configuration 19.

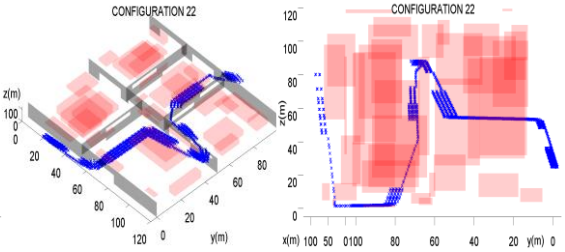


(j) Formation trajectories within configuration 20.

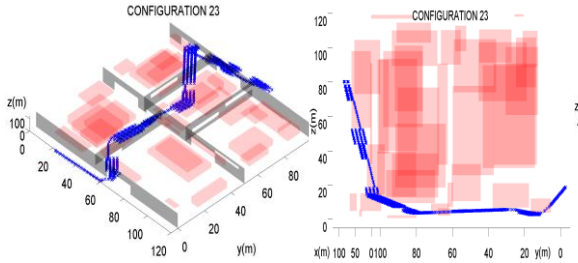
Fig. 5.40. MA-Formation Indoor 1: Multi-agent formation configuration 11-20 for the final generation.



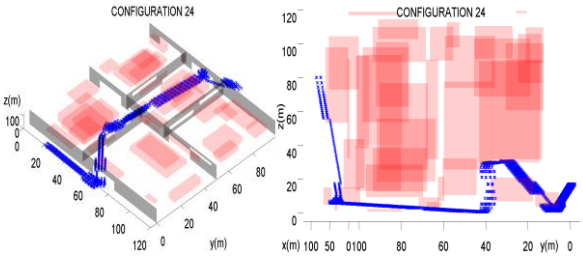
(a) Formation trajectories within configuration 21.



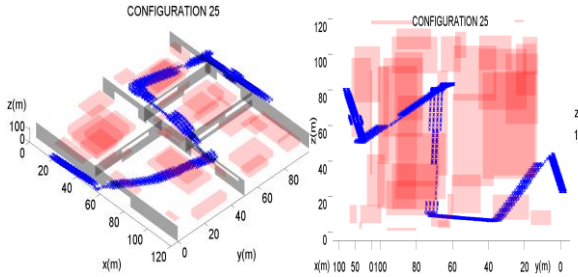
(b) Formation trajectories within configuration 22.



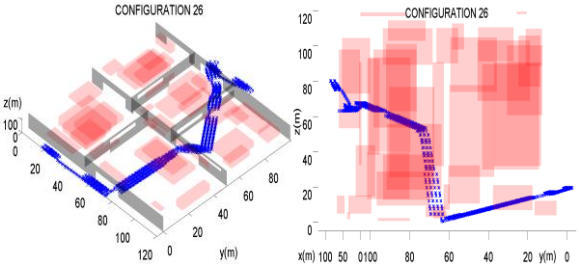
(c) Formation trajectories within configuration 23.



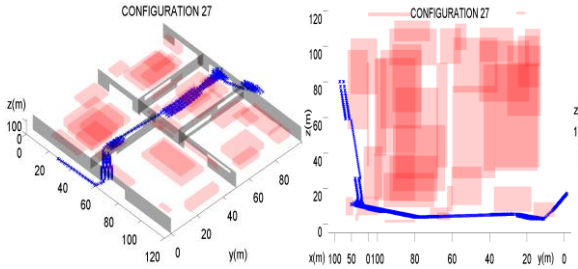
(d) Formation trajectories within configuration 24.



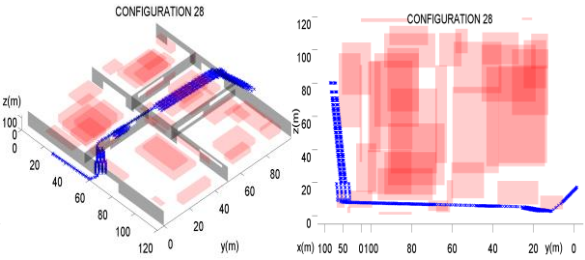
(e) Formation trajectories within configuration 25.



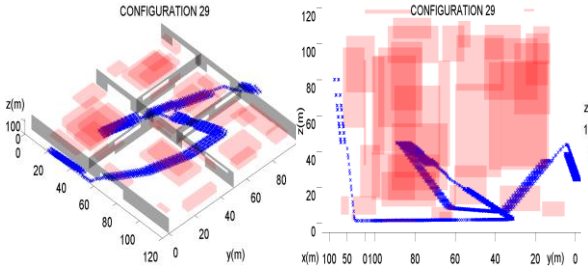
(f) Formation trajectories within configuration 26.



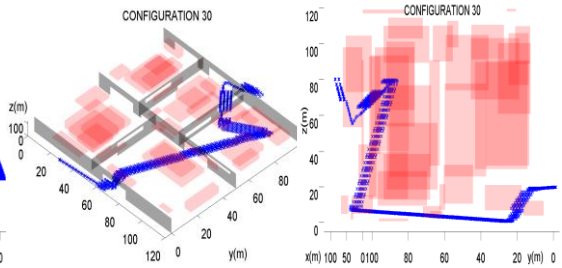
(g) Formation trajectories within configuration 27.



(h) Formation trajectories within configuration 28.



(i) Formation trajectories within configuration 29.



(j) Formation trajectories within configuration 30.

Fig. 5.41. MA-Formation Indoor 1: Multi-agent formation configuration 21-30 for the final generation.

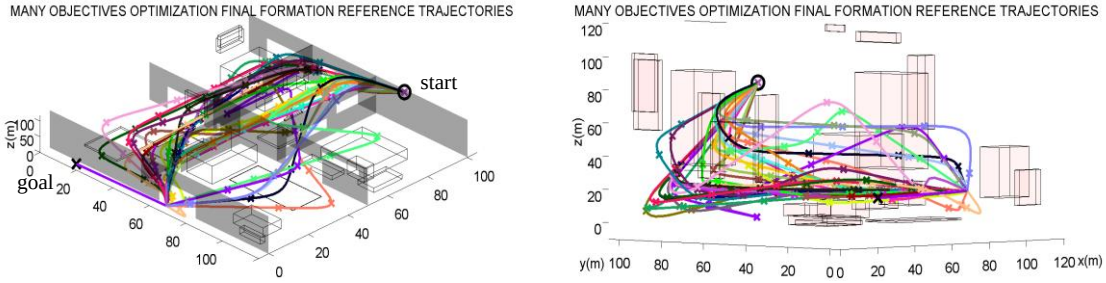
TABLE 5.27. MANY-OBJECTIVES VALUES OF FINAL GENERATION MA-Formation SOLUTIONS WITHIN HIGHLY CLUTTERED INDOOR 1

	PATH LENGTH (*meters*)	ALTITUDE COST (*meters*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	Formation SCALING (ratio)	SAFETY RANGE (*no breach*)	Formation DESIGN (*no variations*)	Formation MAINTAINANCE (*no violations*)	Formation RISE TIME (*sec*)
1	2814.99	39.79	1034.00	52.12	0.24	2.11	4887.58	4.25	8029.00	7.00	2.00	24.67
2	2328.44	9.96	823.00	42.01	0.28	2.09	4303.48	5.75	10808.00	10.00	3.00	34.56
3	2593.69	15.65	951.00	43.67	0.28	3.01	5223.73	5.92	12010.00	10.00	5.00	39.80
4	2551.88	34.65	675.00	52.75	0.24	2.83	4687.25	3.50	6152.00	7.00	3.00	27.38
5	2940.19	15.76	1531.00	53.53	0.26	4.21	6980.92	6.92	12827.00	12.00	4.00	42.31
6	2261.12	34.92	752.00	46.14	0.24	2.37	4351.05	4.17	6232.00	8.00	4.00	29.17
7	3325.23	25.79	1099.00	58.70	0.31	2.85	5486.34	7.83	13799.00	15.00	4.00	61.47
8	2398.95	15.97	910.00	50.93	0.26	2.52	5190.94	6.17	13217.00	11.00	3.00	43.96
9	2887.90	37.42	1485.00	65.27	0.27	3.10	5743.11	7.25	13405.00	13.00	5.00	52.03
10	2614.50	26.28	819.00	56.75	0.28	3.28	5409.61	6.25	7084.00	12.00	3.00	40.95
11	2211.27	9.91	882.00	41.36	0.26	2.65	4465.18	5.42	11976.00	9.00	3.00	31.39
12	3305.67	12.24	1731.00	57.51	0.24	4.68	7540.63	7.00	14131.00	14.00	5.00	48.84
13	2452.57	41.36	998.00	49.22	0.24	2.56	4752.95	4.50	6547.00	8.00	4.00	30.42
14	2977.96	34.10	1547.00	48.95	0.23	3.62	7550.83	5.50	17146.00	11.00	4.00	46.35
15	2936.27	28.10	882.00	51.20	0.28	2.47	4874.77	4.75	14141.00	8.00	3.00	30.80
16	3725.31	26.15	2332.00	82.97	0.25	5.75	8171.98	11.17	12406.00	18.00	8.00	68.10
17	2912.50	43.74	1372.00	60.03	0.23	4.07	6380.42	6.25	9717.00	10.00	4.00	39.00
18	2984.19	26.64	1164.00	52.28	0.26	4.05	6428.26	6.42	20190.00	12.00	5.00	44.42
19	3153.72	25.63	623.00	52.72	0.28	2.98	5315.24	6.33	13129.00	11.00	5.00	48.35
20	4765.83	26.89	1809.00	83.53	0.26	5.66	9053.17	10.25	15734.00	18.00	8.00	75.14
21	2375.88	14.13	744.00	41.49	0.24	3.02	4737.46	4.75	13336.00	8.00	3.00	28.75
22	3511.39	46.97	1429.00	67.26	0.27	8.81	14009.35	6.83	14119.00	13.00	6.00	52.32
23	2664.33	17.89	848.00	56.97	0.28	3.23	5561.15	8.75	17886.00	14.00	5.00	51.33
24	2826.59	14.78	968.00	53.74	0.33	3.41	5695.03	7.00	13719.00	12.00	4.00	44.37
25	3425.18	48.79	1427.00	65.71	0.26	6.99	10159.35	7.58	12713.00	15.00	7.00	58.43
26	2767.86	36.72	695.00	54.34	0.30	3.11	5453.25	9.42	11393.00	17.00	8.00	74.52
27	2184.48	13.49	716.00	42.89	0.26	2.87	4464.23	5.92	13005.00	11.00	3.00	45.72
28	2109.07	13.35	679.00	35.93	0.22	2.65	4496.40	4.25	10542.00	7.00	2.00	27.86
29	3697.76	19.96	1945.00	68.31	0.23	4.97	7682.65	6.25	12838.00	11.00	5.00	40.82
30	2873.60	28.16	1266.00	60.20	0.25	3.35	5853.55	5.50	6281.00	11.00	4.00	41.59

5.7.3 TYPES OF SIMULATION MODELS

This section compares three variations of the highly cluttered indoor environments. The differences between each environment is similar to the MA-Spread application in Table. The results of the first cityscape environment was shown in Section 5.7.1-5.7.2. The simulations that were performed with Indoor 2 and 3 prioritized lower running time to test the abilities of the algorithm to plan paths across a variety of test spaces.

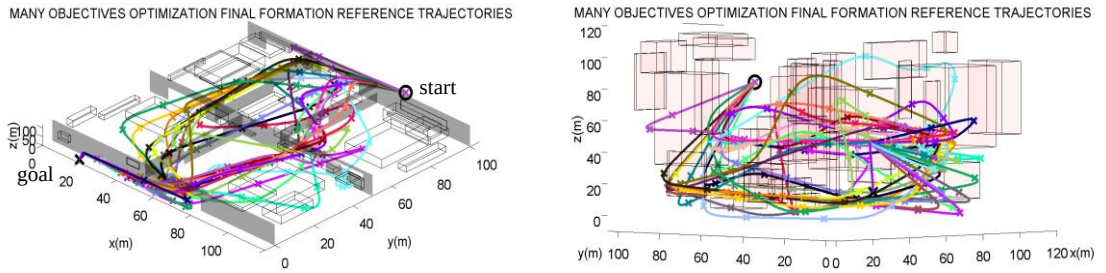
The results for Indoor 2 and 3 are shown in Table 5.28-5.29. Figure 5.42-5.43 shows that the algorithm is able to generate smooth formation reference trajectories across all indoor environments. The data shows that the algorithm is able to minimize 10 of the objectives at a larger percentage in comparison to Indoor 1. Similar to the other environments, these lower resolution formation reference paths will produce a larger spline deviation error which can be improved with a higher sampling rate like Indoor 1.



(a)Top view of formation reference paths within final generation. (b) Side view of formation reference paths within final generation.
Fig. 5.42. MA-Formation Indoor 2: Final generation's multi-agent unique formation reference trajectories.

TABLE 5.28: MA-Formation HIGH RISE INDOOR 2: EXPERIMENTAL RESULTS

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE (*no free) space grids*	SENSORY OVERLAP (*no grid block) overlap*
1	max	7714.15	62.04	2961.00	87.74	2.45	4.42	8338.69	12589.00	13.92	27.00	12.00	125.35
	mean	4667.89	42.74	1636.07	58.58	0.76	2.58	4774.93	5809.93	6.06	11.90	4.87	61.05
	min	1356.00	21.63	102.00	14.31	0.24	0.26	550.32	714.00	0.33	1.00	0.00	4.08
78	max	2403.25	59.48	279.00	21.83	2.62	2.19	1063.02	2061.00	1.50	3.00	2.00	23.85
	mean	1381.57	32.82	106.10	14.39	1.66	1.05	601.18	1012.63	0.50	1.17	0.10	5.47
	min	642.85	17.00	39.00	8.89	1.02	0.34	424.39	322.00	0.00	1.00	0.00	0.00
	%	70.40	23.21	93.51	75.44	-118.42	59.14	87.41	82.57	91.75	90.17	97.95	91.04



(a) Top view of formation reference paths within final generation. (b) Side view of formation reference paths within final generation.
Fig. 5.43. MA-Formation Indoor 3: Final generation's multi-agent unique formation reference trajectories.

TABLE 5.29: MA-Formation HIGH RISE INDOOR 3: EXPERIMENTAL RESULTS

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION *(m)*	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE *(no free) space grids*	SENSORY OVERLAP (*no grid block) overlap*
1	max	6799.37	71.35	2667.00	85.28	1.64	4.01	7179.17	14201.00	12.33	21.00	10.00	96.09
	mean	4110.06	54.28	1431.63	50.93	0.75	1.99	3852.27	7057.10	5.32	9.67	2.90	48.09
	min	1627.59	30.71	115.00	17.52	0.27	0.23	584.36	863.00	0.50	1.00	0.00	4.34
95	max	3395.39	63.00	395.00	33.01	2.38	3.73	1635.93	2109.00	2.00	4.00	1.00	29.29
	mean	1915.91	35.66	190.00	20.11	1.69	1.32	762.49	1248.30	0.79	1.63	0.20	10.09
	min	1253.74	17.56	98.00	13.56	1.02	0.76	568.65	647.00	0.00	1.00	0.00	0.00
	%	53.38	34.30	86.73	60.51	-125.33	33.49	80.21	82.31	85.15	83.14	93.10	79.02

Table 5.30 shows the running time per generation for all three indoor environments that have different parameters. There is one major difference between the results of the indoor and cityscape test space. Here, the formation planning across the indoor environment requires the most processing time. The generation of the free space contour across all three indoor variations uses a large percentage of time. Indoor 1 uses the most amount of total time since it has the largest amount of clutter.

Like most of the simulations, the lower resolution formation paths in Indoor 2 and 3 have a much lower running time than Indoor 1. These results show that these parameters can be changed to influence the performance of the algorithm.

TABLE 5.30: MA-Formation HIGHLY CLUTTERED INDOOR RUNNING TIME.

	ALGORITHM	INDOOR 1 (sec)		INDOOR 2 (sec)		INDOOR 3 (sec)	
MA-FORMATION	GA + MA-RRF Repair + DRMOO	5231.72		951.05		1019.16	
	GA	1.01	0.02 %	0.26	0.03 %	0.36	0.04 %
	MA-RRF Repair	90.07	1.72 %	36.81	3.87 %	158.88	15.59 %
	Identify nearest obstacle sample node	2621.95	50.12 %	338.72	35.62 %	301.19	29.55 %
	Generate free space surface	1045.53	19.98 %	175.35	18.44 %	171.39	16.82 %
	PD Control System	528.79	10.11 %	218.27	22.95 %	178.84	17.55 %
	Formation objectives	426.61	8.15 %	56.05	5.89 %	69.25	6.79 %
	Collision Check	256.44	4.90 %	122.05	12.83 %	282.07	27.67 %

5.8. MA- FORMATION ACROSS A MOUNTAINOUS TERRAIN

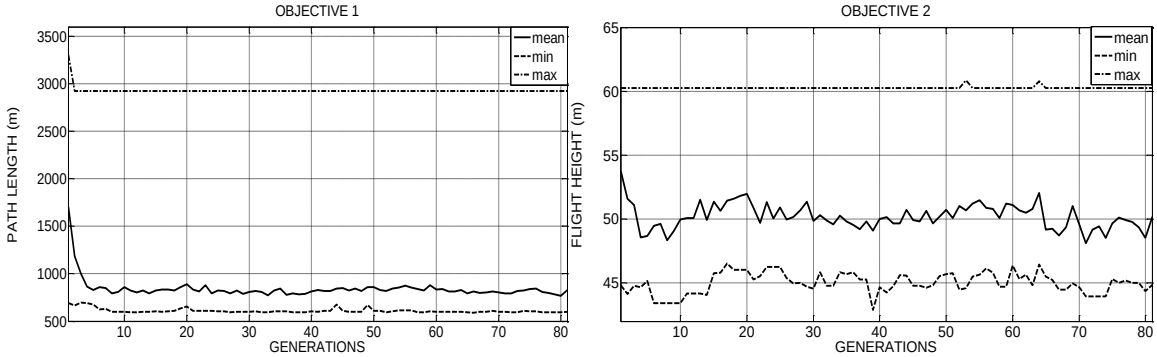
Lastly, this section presents the multi-agent quadrotor formation paths and their shapes across the mountainous terrain. Similar to the indoor environment, this terrain challenges the formation planner in terms of obstacle avoidance. Here, the number of obstacles is significantly higher than the other test spaces. The gradual mountain peaks that are placed all around the terrain can pose a threat to the quadrotors that are in the outer columns of the formation structure. It is important that the formation planner is capable of generating well minimized paths that do not collide with any peaks.

The different trade-off values for the 12 objective functions are evaluated in this section. Firstly, the progression of the objectives across generations for Mountain 1 is shown in Figure 5.44-5.45. Then, the minimization of the cost functions for the formation trajectories in the final generation is shown in Table 5.31. The unique formation reference paths for the agents are presented in Figure 5.46. Lastly, the images of each formation configuration within the final population are shown in Figure 5.47-5.49. The cost trade-off values for each configuration are shown in Table 5.32.

5.8.1. MANY OBJECTIVES VALUES

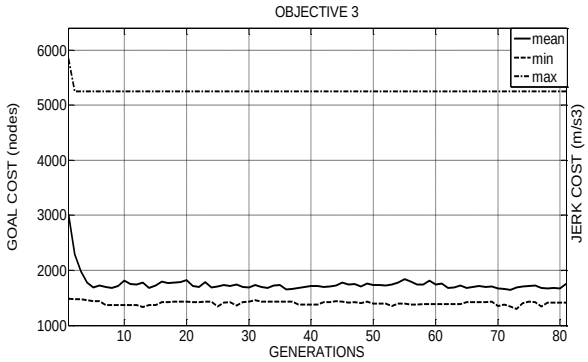
The mean, max and min values of all 12 objectives are shown below. Many of the objectives reach its minimum point after the 80^{th} generation. One obvious pattern that exists in most objectives is the maintenance of the max value across 81 generations. Analysis of the formation trajectory population shows that one path has been maintained through many iterations. This path provides the maximum values for most of the objective functions. The reason it was kept despite its disadvantages is because it is highly diverse in comparison to the rest of the population.

The average values for the formation configuration population is shown in Table 5.31. The values of three objectives are reduced at less than 10%. The number of nodes at higher parts of the mountainous terrain is lowered by 7%. Both the spline deviation error and number of safety zone breaches are minimized by 5%. All of the formation-based costs are minimized by less than 40%. The values of the rest of the many cost functions are reduced by more than 40% in comparison to the initial MA-RRF path population. The formation trajectories are 51% shorter in length. Similarly, the paths are more direct since the number of goal node deviations is lessened by 42%. The agents in formation are required to perform less aggressive turns with a reduction in jerk value of 50%. Lastly, the quadrotors are able to fly at a more than 69% optimal speed as well as less than 61% fuel consumption.

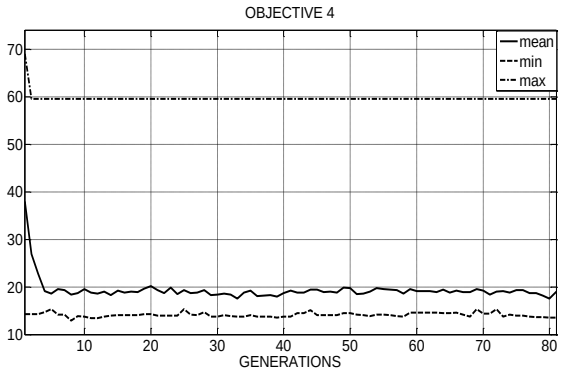


(a) Path length objective values across generations.

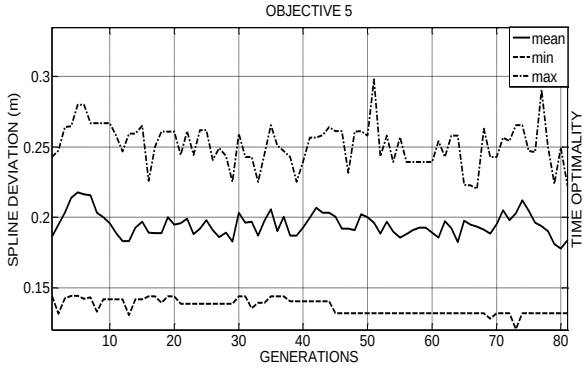
(b) Flight height objective values across generations.



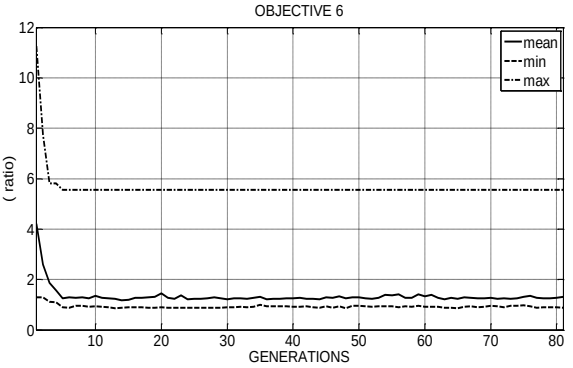
(c) Goal deviations objective values across generations.



(d) Jerk cost objective values across generations

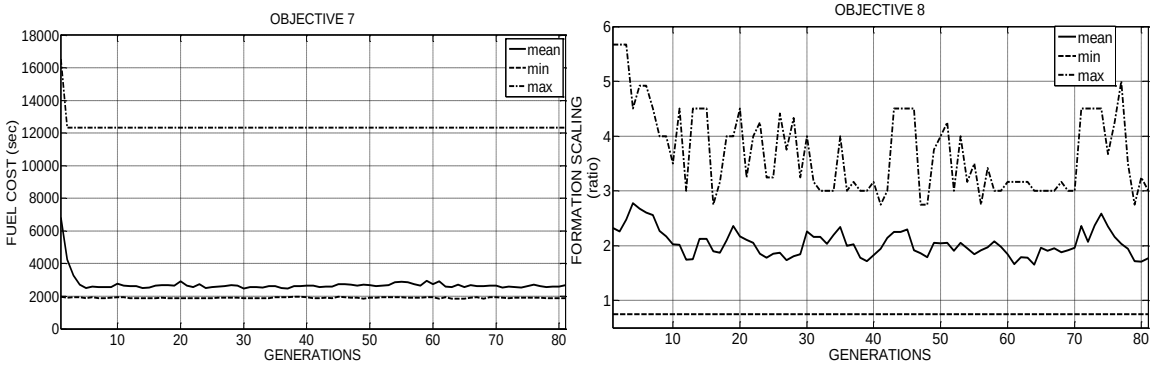


(e) Spline deviation error objective values across generations.

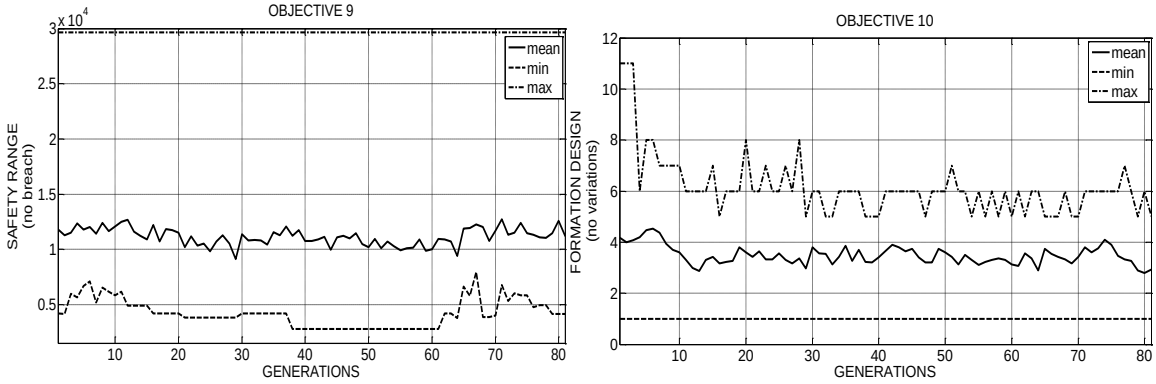


(f) Time optimality objective values across generations.

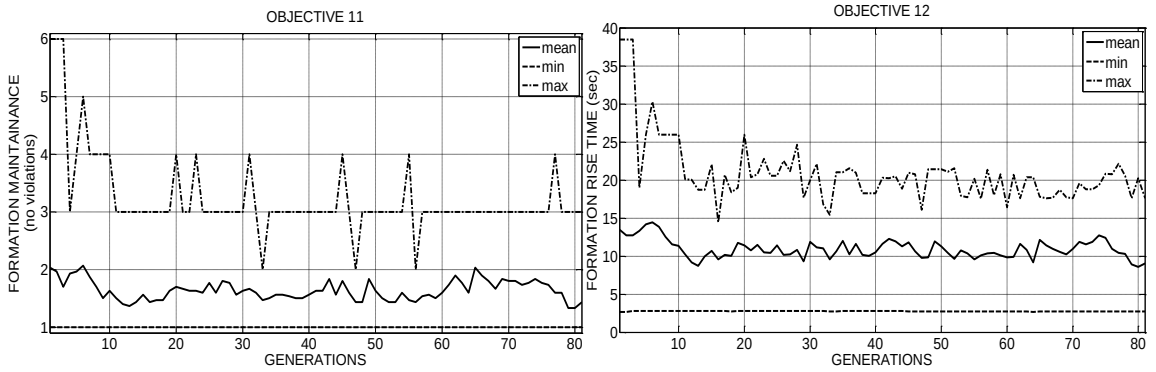
Fig. 5.44. MA-Formation Mountain 1: Progression of the objectives 1-6 across generations.



(a) Flight time objective values across generations. (b) Formation scaling ratio objective values across generations.



(c) Safety zone breaches objective values across generations. (d) Formation changes objective values across generations.



(e) Formation maintenance objective values across generations. (f) Formation rise time objective values across generations.

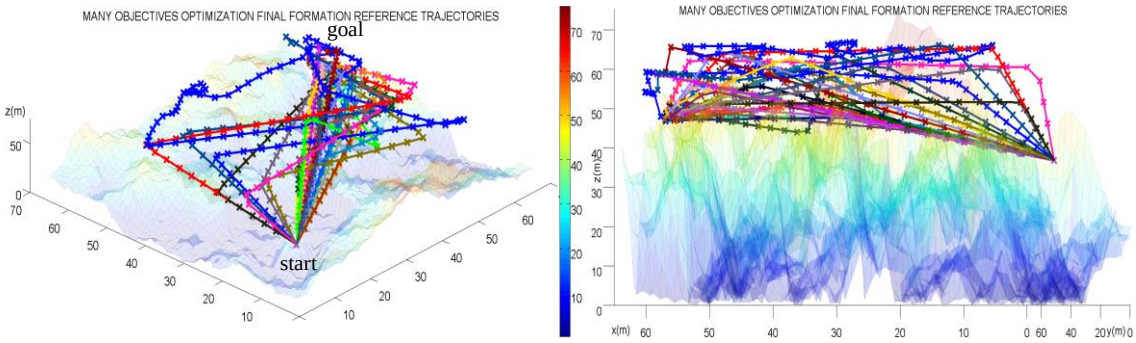
Fig. 5.45. MA-Formation Mountain 1: Progression of the objectives 5-12 across generations.

TABLE 5.31: MA-Formation MOUNTAINOUS TERRAIN 1 EXPERIMENTAL RESULTS

ITER		PATH LENGTH (*meters*)	ALTITUDE COST (*meters*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	SAFETY RANGE (*no breach*)	Formation SCALING *(ratio)*	Formation DESIGN (*no variations*)	Formation MAINTAINANCE (*no violations*)	Formation RISE TIME (*sec*)
1	max	3303.10	60.27	5844.00	68.98	0.24	11.25	16523.00	29666.00	5.67	11.00	6.00	38.44
	mean	1698.50	53.74	3018.10	37.95	0.19	4.21	6821.60	11793.00	2.32	4.17	2.03	13.47
	min	692.93	44.83	1488.00	14.27	0.14	1.29	1983.30	4187.00	0.75	1.00	1.00	2.69
81	max	2924.90	60.27	5250.00	59.51	0.22	5.55	12318.00	29666.00	3.00	5.00	3.00	17.61
	mean	831.16	50.21	1759.20	18.96	0.18	1.30	2681.20	11172.00	1.78	2.93	1.43	9.03
	min	597.27	44.85	1411.00	13.53	0.13	0.88	1892.70	4128.00	0.75	1.00	1.00	2.70
	%	51.07	6.57	41.71	50.04	5.26	69.12	60.70	5.27	23.28	29.74	29.56	32.96

5.8.2. TRAJECTORY POPULATION

The unique formation reference paths for Mountain 1 are displayed in different colours within Figure 5.46. The image shows that each formation reference trajectory has nodes all across the terrain. The trajectories that are designed for the mountainous terrain are more simplistic and less diverse than the paths within the other test environments. This is due to the large amount of free space between the mountain peaks. The initial paths before path pruning are diverse in direction. The path pruning process removes redundant nodes and causes the paths to converge towards the centre of the large open space. Due to this, unlike other test environment, many paths share the minimal values for certain objectives. Still, the algorithm succeeds in free space mapping and avoiding all of the small peak changes.

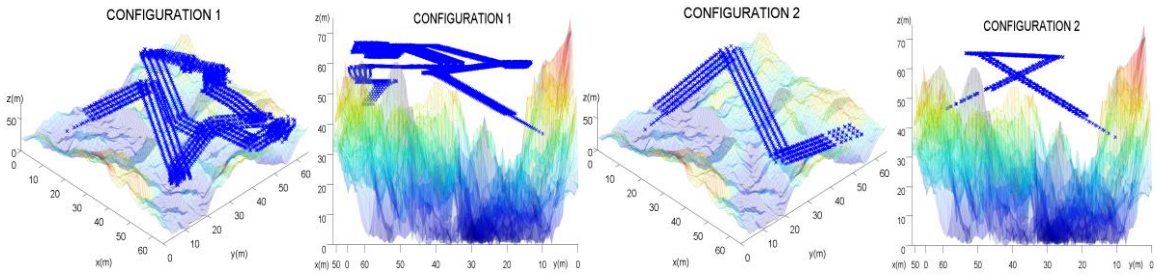


(a) Top view of formation reference paths within final generation. (b) Side view of formation reference paths within final generation.

Fig. 5.46. MA-Formation Mountain 1: Final generation's multi-agent unique formation reference trajectories.

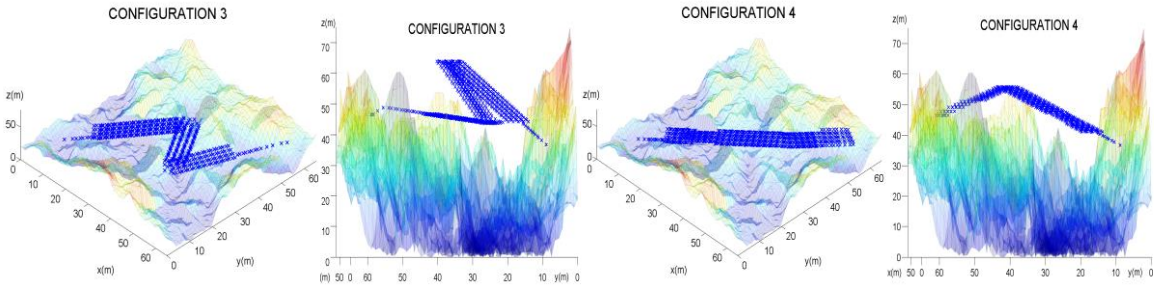
Lastly, the formation paths and their shapes for the mountainous terrain are shown in Fig.5.47-5.49. Firstly, formation path 30 has the minimal value for the path length. Fig 5.49(j) shows that the paths are very direct and short in comparison to the other options. The option with the most nodes at lower parts of the terrain is configuration 12. The data can be seen and confirmed by Fig 5.48(b). Choice 4 has the lowest value for the goal deviation cost. On the other hand, option 29 has the minimal jerk cost. Next, configuration 2 as shown in Fig 5.47(b) has the minimum value for both the spline deviation error and formation rise time. It can be seen that the agents will only have to transition between two shapes and fly across an easy to follow path.

Option 6 produces paths that allow the quadrotors to fly at close to ideal velocity whereas choice 8 requires minimal fuel consumption. Configuration 1 is displayed in Fig 5.47(a). This path is lengthy in comparison to the other paths. The reason that this path was maintained across generations is due to its diversity in direction. It also has the minimal value for the formation shape scaling ratio. Lastly, formation configuration 5 has the lowest values for the number of safety zone breaches, formation shape changes and maintenance. Fig 5.47(e) shows that the agents will only have to morph into two different shapes. The data that is presented within this section will aid the end user in their final choice of formation trajectory.



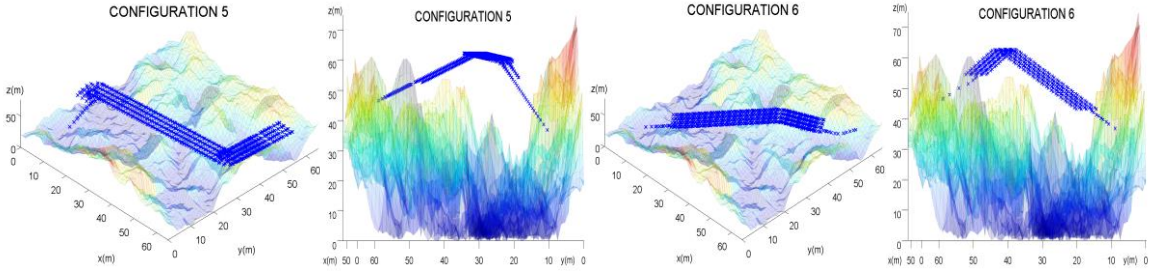
(a) Formation trajectories within configuration 1.

(b) Formation trajectories within configuration 2.



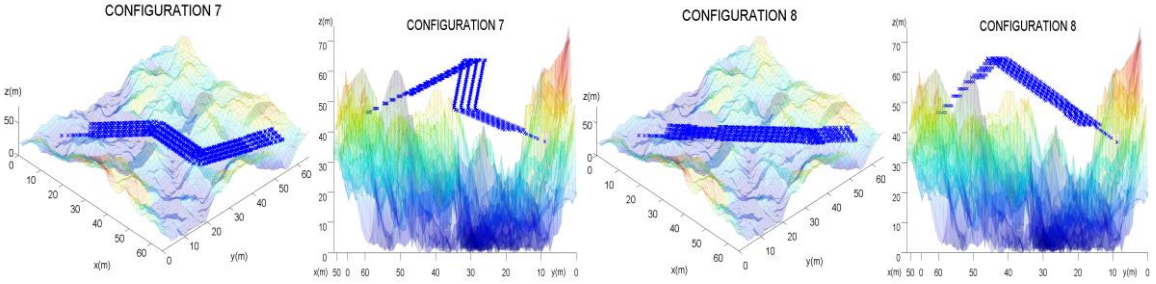
(c) Formation trajectories within configuration 3.

(d) Formation trajectories within configuration 4.



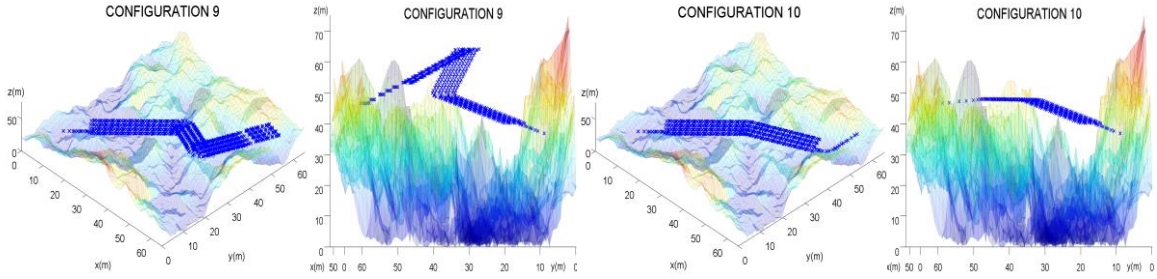
(e) Formation trajectories within configuration 5.

(f) Formation trajectories within configuration 6.



(g) Formation trajectories within configuration 7.

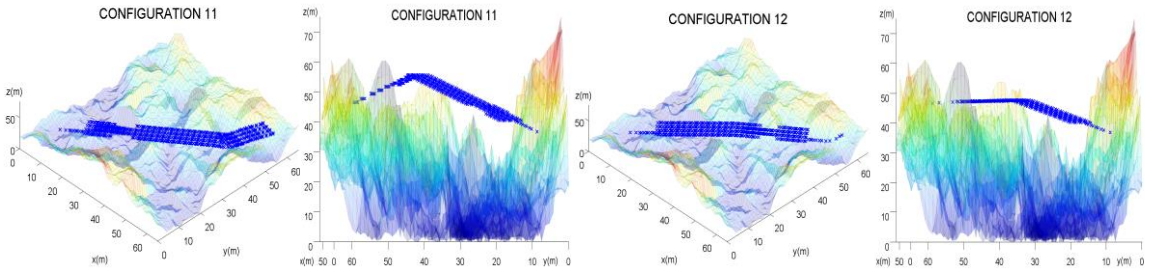
(h) Formation trajectories within configuration 8.



(i) Formation trajectories within configuration 9.

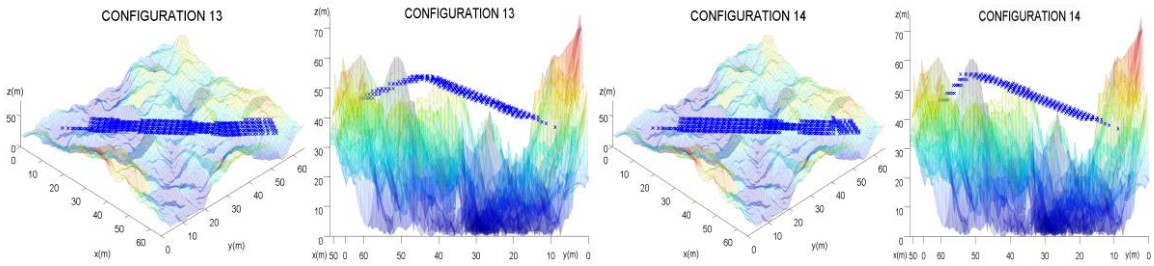
(j) Formation trajectories within configuration 10.

Fig. 5.47. MA-Formation Mountain 1: Multi-agent formation configuration 1-10 for the final generation.



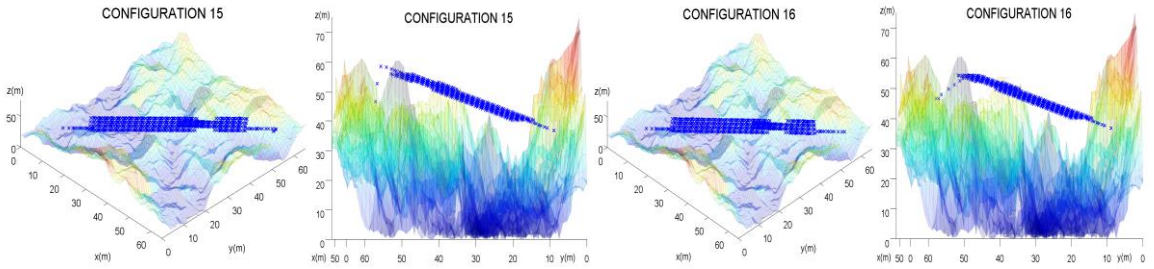
(a) Formation trajectories within configuration 11.

(b) Formation trajectories within configuration 12.



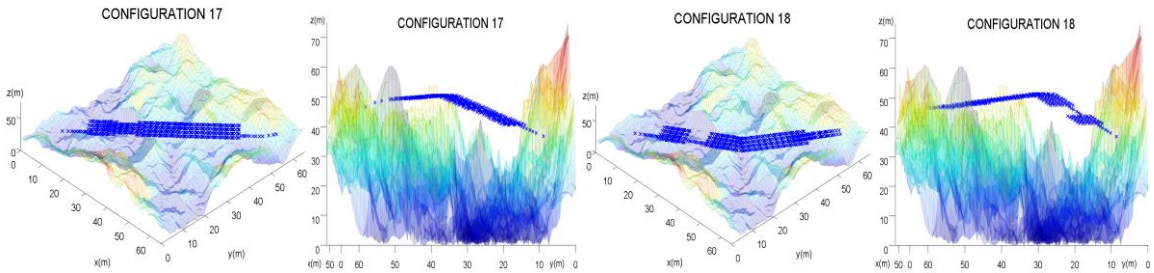
(c) Formation trajectories within configuration 13.

(d) Formation trajectories within configuration 14.



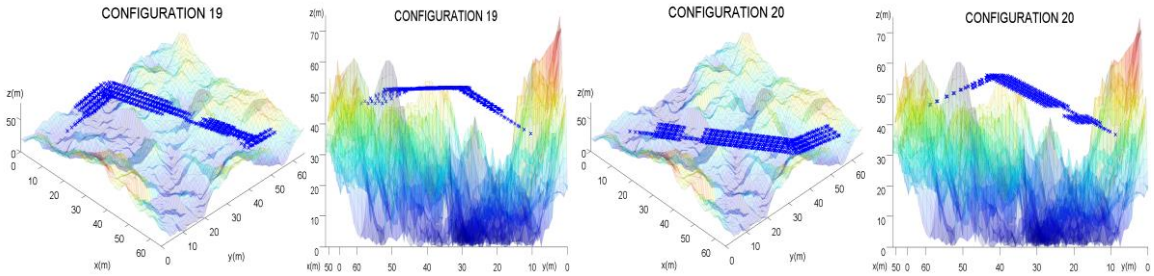
(e) Formation trajectories within configuration 15.

(f) Formation trajectories within configuration 16.



(g) Formation trajectories within configuration 17.

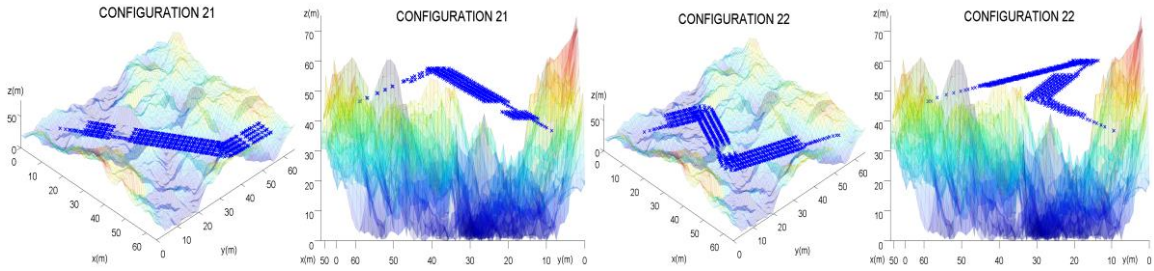
(h) Formation trajectories within configuration 18.



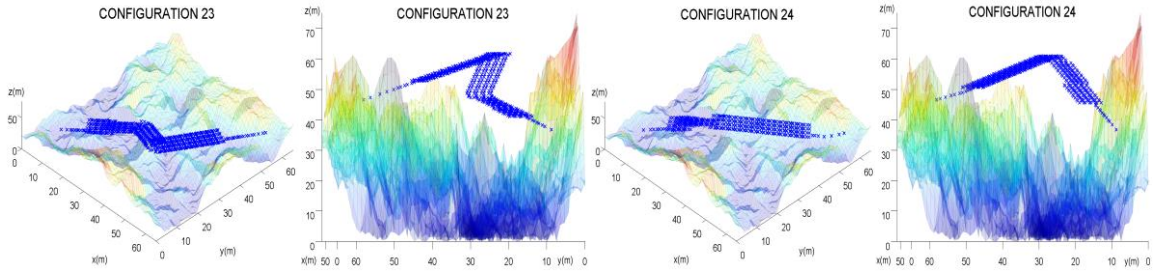
(i) Formation trajectories within configuration 19.

(j) Formation trajectories within configuration 20.

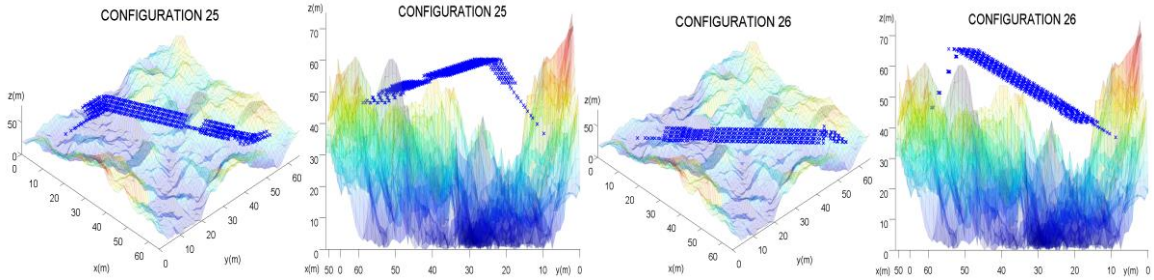
Fig. 5.48. MA-Formation Mountain 1: Multi-agent formation configuration 11-20 for the final generation.



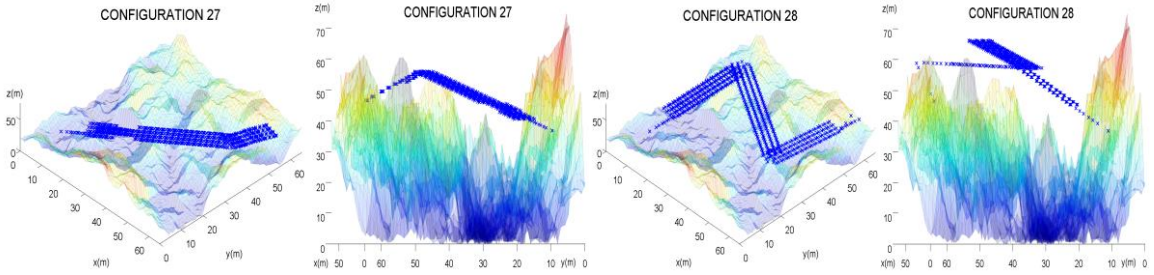
(a) Formation trajectories within configuration 21. (b) Formation trajectories within configuration 22.



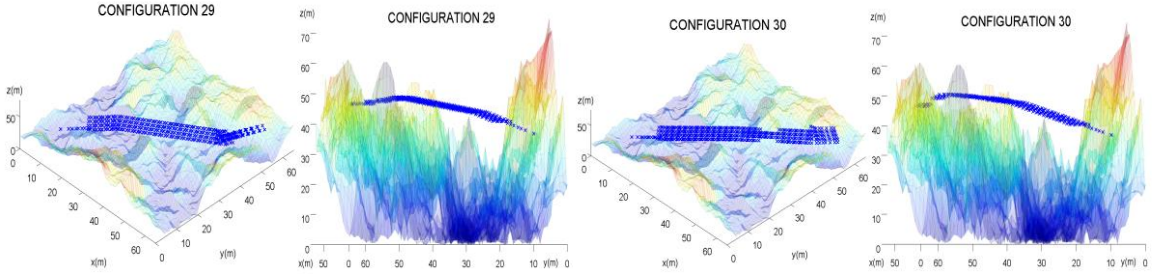
(c) Formation trajectories within configuration 23. (d) Formation trajectories within configuration 24.



(e) Formation trajectories within configuration 25. (f) Formation trajectories within configuration 26.



(g) Formation trajectories within configuration 27. (h) Formation trajectories within configuration 28.



(i) Formation trajectories within configuration 29. (j) Formation trajectories within configuration 30.

Fig. 5.49. MA-Formation Mountain 1: Multi-agent formation configuration 21-30 for the final generation.

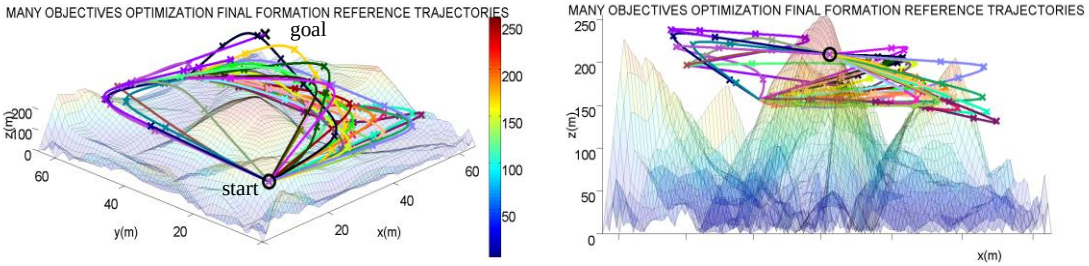
TABLE 5.32. MANY-OBJECTIVES VALUES OF FINAL GENERATION MA-Formation SOLUTIONS WITHIN MOUNTAINOUS TERRAIN 1

	PATH LENGTH (*meters*)	ALTITUDE COST (*meters*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio)*	FUEL COST (*sec*)	Formation SCALING (ratio)	SAFETY RANGE (*no breach*)	Formation DESIGN (*no variations*)	Formation MAINTAINANCE (*no violations*)	Formation RISE TIME (*sec*)
1	2924.93	60.27	5250.00	59.51	0.14	5.55	12318.18	0.75	29666.00	1.00	1.00	2.79
2	1105.06	58.57	2280.00	23.41	0.13	1.99	4743.08	0.75	8207.00	1.00	1.00	2.70
3	951.62	49.69	1653.00	23.70	0.18	1.39	2831.75	1.50	9109.00	2.00	1.00	6.74
4	614.41	48.36	1411.00	16.64	0.18	0.99	2018.02	1.25	9521.00	3.00	2.00	9.95
5	876.16	58.23	1859.00	17.45	0.15	1.15	2870.64	0.75	4128.00	1.00	1.00	3.07
6	713.92	52.29	1509.00	16.75	0.17	0.88	1895.92	1.50	6869.00	2.00	1.00	5.57
7	769.73	51.16	1687.00	17.78	0.15	1.03	2329.84	0.75	9180.00	1.00	1.00	2.73
8	752.21	52.52	1421.00	18.55	0.19	0.89	1892.70	1.25	5644.00	3.00	2.00	10.19
9	760.52	50.62	1723.00	17.98	0.14	1.12	2478.31	0.75	11033.00	1.00	1.00	2.72
10	636.00	45.22	1645.00	13.85	0.16	1.11	2219.50	1.50	14434.00	2.00	1.00	5.39
11	690.47	47.71	1549.00	16.31	0.18	1.07	2117.08	1.75	11661.00	3.00	1.00	9.05
12	705.04	44.85	1567.00	14.39	0.21	1.04	1934.22	2.50	15188.00	4.00	1.00	11.53
13	656.59	47.09	1425.00	17.06	0.22	1.02	2029.97	1.75	11629.00	5.00	3.00	17.61
14	613.38	47.31	1437.00	14.46	0.18	1.03	2025.87	1.75	11550.00	3.00	1.00	8.88
15	713.76	48.48	1612.00	15.35	0.20	1.09	2029.27	2.50	10644.00	4.00	1.00	11.63
16	711.69	46.67	1540.00	15.03	0.21	1.06	1947.43	2.50	12823.00	4.00	1.00	11.78
17	685.16	46.25	1579.00	16.38	0.21	1.08	2040.63	2.00	13261.00	4.00	2.00	12.58
18	710.73	46.58	1564.00	16.31	0.22	0.99	1951.60	2.75	12211.00	4.00	2.00	12.04
19	822.71	49.69	1791.00	17.77	0.21	1.35	2666.35	2.75	10261.00	5.00	2.00	14.20
20	687.28	48.10	1582.00	16.79	0.18	1.07	2109.43	2.25	12823.00	3.00	2.00	9.30
21	692.90	49.13	1602.00	17.41	0.18	1.36	2665.32	2.25	12100.00	3.00	2.00	9.39
22	972.52	51.53	2074.00	21.39	0.20	1.39	2785.92	3.00	12096.00	4.00	2.00	13.53
23	830.35	50.77	1677.00	21.05	0.20	1.27	2408.46	2.00	11055.00	4.00	2.00	13.36
24	768.07	52.27	1456.00	19.43	0.22	1.20	2195.93	2.50	7056.00	4.00	1.00	12.48
25	806.16	53.85	1516.00	17.95	0.22	1.03	2075.27	2.75	5984.00	4.00	2.00	12.80
26	716.47	52.65	1557.00	17.84	0.19	0.97	1940.41	1.25	5806.00	3.00	2.00	10.18
27	688.09	47.81	1551.00	16.38	0.18	1.09	2110.20	1.75	11428.00	3.00	1.00	9.01
28	1146.71	57.58	2217.00	24.71	0.15	1.82	3597.32	1.50	10734.00	2.00	1.00	5.53
29	614.79	45.12	1598.00	13.53	0.16	1.07	2186.60	1.25	14760.00	2.00	1.00	5.37
30	597.27	45.78	1443.00	13.70	0.18	1.05	2021.15	1.75	14298.00	3.00	1.00	8.80

5.8.3 TYPES OF SIMULATION MODELS

This section compares three variations of the mountainous terrain test space. The differences between each environment is similar to the MA-Spread application. The results of the first terrain was shown in Section 5.8.1-5.8.2. The simulations that were performed with Terrain 2 and 3 prioritizes lower running time to quickly test the abilities of the algorithm to plan paths across a variety of test spaces.

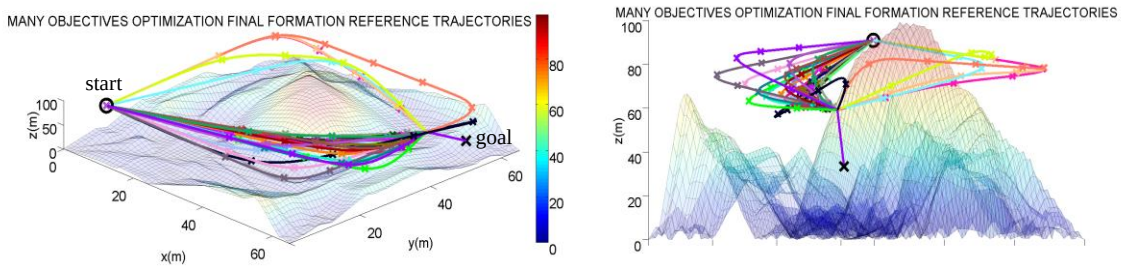
The data or Terrain 1 and 2 is shown in Table 5.33-5.34. The final formation reference trajectories are also displayed in Figure 5.50-5.51. The images show that the formation planes are able to design path nodes that avoid all mountain peaks successfully. This was done by identifying the small and large peaks with accuracy. The values of ten objectives are well minimized across generations. There is a slight increase in altitude cost and spline deviation error. These results show that the algorithm is capable of adapting to various terrains.



(a) Top view of formation reference paths within final generation. (b) Side view of formation reference paths within final generation.
Fig. 5.50. MA-Formation Mountain 2: Final generation's multi-agent unique formation reference trajectories.

TABLE 5.33: MA-Formation HIGH RISE MOUNT 2: EXPERIMENTAL RESULTS.

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE (*no free*) *space grids*	SENSORY OVERLAP (*no grid block*) *overlap*
1	max	4924.62	218.80	2131.00	79.87	2.17	5.91	5513.61	10621.00	11.42	34.00	21.00	220.70
	mean	2152.87	182.17	1027.43	30.23	0.82	2.59	2443.22	3328.90	5.59	14.47	6.83	91.96
	min	578.19	120.12	31.00	4.28	0.21	0.14	157.96	477.00	1.67	5.00	1.00	32.33
84	max	1489.84	227.98	311.00	17.92	2.21	3.89	551.64	1841.00	6.67	15.00	8.00	117.99
	mean	1037.48	184.15	129.47	11.10	1.35	1.81	374.30	1021.33	4.24	9.67	3.57	71.06
	min	511.88	150.67	5.00	2.76	0.63	0.29	71.85	72.00	0.50	1.00	0.00	7.19
	%	51.81	-1.09	87.40	63.28	-64.63	30.06	84.68	69.32	24.15	33.17	47.73	22.73



(a) Top view of formation reference paths within final generation. (b) Side view of formation reference paths within final generation.
Fig. 5.51. MA-Formation Mountain 3: Final generation's multi-agent unique formation reference trajectories.

TABLE 5.34: MA-Formation HIGH RISE MOUNT 3: EXPERIMENTAL RESULTS.

ITER		PATH LENGTH (*m*)	ALTITUDE COST (*m*)	GOAL COST (*nodes*)	JERK COST (m/s^3)	SPLINE DEVIATION (*m*)	TIME OPTIMALITY (*ratio*)	FUEL COST (*sec*)	NETWORK TOPOLOGY (*no loss*)	COLLISION AVOIDANCE (*no collisions*)	SAFETY RANGE (*no breach*)	TERRAIN COVERAGE *(no free) space grids*	SENSORY OVERLAP (*no grid block) overlap*
1	max	3436.49	83.81	2337.00	50.31	1.69	11.37	4985.10	12889.00	9.67	28.00	14.00	191.51
	mean	1765.44	69.15	1192.97	24.74	0.56	4.53	2530.76	3593.73	3.05	9.00	4.50	54.42
	min	480.30	40.67	64.00	2.69	0.19	0.11	71.19	148.00	0.00	1.00	0.00	0.00
56	max	1068.99	82.82	438.00	11.26	1.92	2.46	525.65	1318.00	4.00	10.00	5.00	77.59
	mean	746.27	73.76	164.13	7.21	1.24	1.17	278.21	576.93	2.09	5.53	1.87	38.33
	min	641.20	65.46	73.00	4.79	0.56	0.54	162.06	12.00	0.00	1.00	0.00	0.00
	%	57.73	-6.67	86.24	70.86	-121.43	74.26	89.01	83.95	31.48	38.56	58.44	29.57

Table 5.35 shows the running time for all three variations of the mountainous terrain for the MA-Formation mission. The results here show similarities with the MA-Spread mission across the mountainous terrains. It shows that the collision check for formation flight also requires a large percentage of the running time. One exception is Mountain 2 which uses most of its running time on the parallel run control system for eight agents. It also utilizes 10-20% of its running time on two of the formation planning algorithms.

As with the other simulations, these results show that the algorithm can minimize the values of many objectives. It also shows that fast initial testing can be performed with lower resolution paths.

TABLE 5.35: MA-Formation MOUNTAINOUS TERRAIN RUNNING TIME.

	ALGORITHM	MOUNT 1 (sec)		MOUNT 2 (sec)		MOUNT 3 (sec)	
MA-FORMATION	GA + MA-RRF Repair + DRMOO	2601.70		610.03		700.55	
	GA	4.28	0.16 %	0.42	0.07 %	0.83	0.12 %
	MA-RRF Repair	266.95	10.26 %	23.00	3.77 %	70.65	10.08 %
	Find nearest obstacle sample node	397.14	15.26 %	132.47	21.72 %	140.63	20.07 %
	Generate free space surface	274.01	10.53 %	102.86	16.86 %	83.99	12.00 %
	PD Control System	257.78	9.91 %	214.89	35.23 %	145.49	20.77 %
	Formation objectives	118.94	4.57 %	53.99	8.85 %	36.60	5.22 %
	Collision Check	1417.04	54.47 %	80.80	13.24 %	263.38	37.60 %

5.9 SUMMARY

This final chapter presented the simulation results for the MA-RRF, GA and DRMOO algorithm across two different multi-agent quadrotor UAV missions. This path planner has utilized algorithms that have the advantage of parallel processing. Here, a control system, sampling based planner, evolutionary algorithm and many objectives optimization algorithm are designed to run on any multi-thread system. Thus, the processing time for both missions can be greatly reduced with a more advanced processing system.

The results show that the trajectory planning algorithm has successfully generated a large collection of trajectories for multiple quadrotors within the MA-Spread application. Firstly, the images prove that the path planner is capable of both high and low-resolution

free space mapping. It is also able to extract collision free paths within all three test environments. A large collection of multi-agent paths was generated by MA-RRF and GA. These versions of the basic algorithm have been successfully modified for diverse path generation. Secondly, the findings for MA-Spread show that the resources of all agents have contributed towards each mission. It has optimized a team of quadrotors collectively as opposed to individually. Thirdly, the motions of all four quadrotors were well estimated through the closed-loop control system. Lastly, the results show that the 12 objectives for MA-Spread task were either well minimized or maintained across generations.

Next, the outcome of the MA-Formation application is a multi-layer system that successfully performs many-objectives optimization across 12 different objectives. Initially, environment free space mapping and the creation of free space contours between obstacles was accomplished. Next, the images show that both the MA-RRF and GA are capable of generating a population of reference paths across all test environments. Then, the results show that the dynamic formation planner is capable of designing formation shape for all 8 agents accordingly. Here, the images prove that formation planner can work in high resolution. This level of resolution is advantageous for the full obstacle avoidance and the design of adaptive formation shapes. Later, the independent trajectories for each agent were applied as input for the multi-agent control. Here, the estimated values of all objectives aided in the ranking of each formation trajectory. Lastly, the many-objectives optimization process was implemented through dimensionality reduction. The final results show that the algorithm is able to find a diverse set of solutions for each scenario. Similarly, all objectives are minimized or maintained without the extreme degradation of one cost over the other. Thus, the end user was provided with a variation of solutions where formation speed, rigidness or simplicity can be used as the final decision factor.

One important consequence of this chapter is that the end user is delivered all of these findings in an easily interpretable manner in order to make a final decision. The data is well organized and the images are in high resolution. All of these results will allow the end user to successfully implement their own post-processing preferences.

CHAPTER 6: CONCLUSION

This research generated a large collection of optimized trajectories for multi-agent quadrotors. The hybridized algorithm extracted trajectories with various trade-off values for all agents without discrimination. This allows the user to check the availability of the resources of all agents to contribute towards the completion of a task. This study created a balance between diverse and optimal solutions through dimensionality reduction. The results showed that the algorithm performs successfully in finding a diverse set of optimal solutions within each environment. The end user is supplied with high resolution visual imagery and well-organized data on the multi-agent quadrotor UAV trajectories. The additional knowledge will assist the end user in making a final choice.

6.1. RESEARCH PURPOSE AND FINDINGS

The main goal of this study was to create a multi-agent quadrotor system of at least three agents. The designed multi-agent quadrotor system must be capable of executing simple tasks cooperatively. As the study progressed, these targets became either more refined or expanded in terms of the study areas that it covered.

Firstly, a variety of cooperative tasks were explored. Some of the multi-agent quadrotor missions that were explored were load lifting, target tracking as well as search and rescue. Designing a separate system for these tasks proved to be time consuming and complex. One of the themes that emerged from these various tasks is that they can be divided into two distinct cooperative missions. This study finds that most multi-agent tasks can be generalized into either spatially spread or formation flight. The purpose of this research was realigned to design a standardized platform for most multi-agent missions. Thus, the end user is awarded a high level of flexibility when implementing this platform towards their desired mission.

Secondly, a mathematical model of the quadrotor must be used to simulate the movements of each agent. In the beginning, an accurate representation of the quadrotor's body and rotor's forces was implemented. An LQR control system was used to simulate the changes in the quadrotor's movements. This study finds that simulating a highly accurate and minimal error system will require a much larger amount of time in comparison to a more simplistic version. This delay is expanded further when multiple agents are simulated simultaneously. It became necessary to find a balance between processing time and accuracy. Here, the simplified model that is often adopted by researchers is run together with a PD control system. Minimal jerk smooth splines are used to transform the paths nodes into time-based trajectories. The purpose of this work evolved towards using a fast, closed-loop control system that can be run in parallel and has minimal deviation error.

Next, it was important to simulate the rotational and translational direction that the quadrotors were flying towards across a test environment. It is important to show through imagery or data that all obstacles are avoided. The visual imagery will be easily understood by users with different levels if knowledge about multi-agent UAVs. The data should show if the agents are capable of reaching their desired end point. In the beginning, a simple 3D plot with cones as obstacles was created to visualize the path that each agent was flying across. It is fast, noncomplex and popular amongst many researchers. Next, this basic environment was expanded towards creating a real-time moving simulation of the multi-agent quadrotors. A high resolution indoor and forest environment was designed. Similarly, accurate 3D model of the quadrotors was designed to run in real-time. This version proved to require a high amount of processing speed. The findings from these two variations are that a balance between high levels of visual animation and complexity must be struck. It is still important that the test environments contain all of the complexities of real life test spaces. The target of this research was changed to generating high resolution imagery that can be studied post-processing. The structures and obstacles within the cityscape, indoor and mountain environment mimic their real-life versions. The images contain all of the important information that is required by the end user. A database also accompanies these images. This database contains knowledge regarding the states of each agent, their exact coordinates and objective values. It also shows if all obstacles are safety avoided.

Now, the main goal of this research was to design a multi-agent free space mapping and path planning system. This planner must be capable of producing feasible paths for all of the quadrotors. Initially, a variety of path planners were implemented. The basic shortest path algorithm, virtual potential function (VPF), leader-follower formation flight and consensus algorithm were all tested. Grid based mapping was utilized for mapping the free space within the simulated test environments. The findings show that there was a tendency for each path planner to perform better in certain tasks. Some planners were suited to spread whereas others were ideal for formation flight. Similarly, some were more suitable for generating one best path whereas the others were capable of extracting many paths. Another issue was that some were too complex for a multi-agent system. Thus, the goal of this research moved towards designing a highly flexible path planner that is a combination of different planners. This planner can be standardized for both spread and formation flight. It must also be able to generate a large collection of possible paths. The initial free space mapping and path planning is quickly performed by the sampling-based MA-RRF planner. Next, a more refined GA path planner is implemented for further generations. A virtual structure formation planner is embedded after the path planning process for formation planning.

Lastly, this research originally aimed to optimize certain objective functions. Many researchers often implement popular objectives such as minimizing path length, error, fuel consumption, agent-to-agent distances and collision avoidance. They consider the environmental and mechanical limitations of their mission. The initial work within this project implemented these objectives with an aggregated cost function. The findings showed that there is a tendency for the minimization of certain objectives to be affected by the others. The team was also required to produce predetermined preferences for the objectives. This can be difficult for inexperienced researchers or drone pilots. The research then expanded towards a more standardized platform that mathematically defines and optimizes many objectives simultaneously. Thus, there was a strong motivation to design a multi-agent system can be applied towards many missions, agents and test environments. Many-objectives optimization is implemented in order to achieve this goal. The optimization of many objectives produces an optimal and diverse collection of paths for the end user.

6.2. RELATIONSHIP WITH PREVIOUS STUDIES

This study is able to provide some insight into trajectory planning for multi-agent quadrotors. Some of the finding within this thesis can be in agreement with previous research publications. On the other hand, there are some findings that can differ from the conclusions of other published studies. The relationship between this thesis and previous studies is presented within this section.

The first stage of any trajectory planning algorithm is the free space mapping process. The MA-RRF sampling-based planner is applied within this research project. This multi-tree algorithm was inspired by the different variations of the RRT planner that is presented in [31]. Study [31] theorises that a multi-tree system can become highly complex. The pseudocode within Chapter 3 confirms that the MA-RRF planner is tougher to implement in comparison to the basic RRT. Still, the multiple trees are capable of fully exploring the three different test environments with speed. The free space mapping and path extraction process is also much faster than running the basic RRT planner sequentially for each quadrotor. One study that applies multi-tree RRT is [49]. In this case, six trees are grown from different locations within the test space. The findings in this research project differ from the results of [49]. Their work requires the multi-tree algorithm to be run multiple times in order to properly interpret the data. The results are aggregated in order to produce short term paths. Chapter 3 of this thesis shows that it is possible to design long term paths with a single run of the multi-tree system.

The authors of [48] also implement a multi-tree RRT system. Their planner includes data sharing between multiple CPUs. In their system, all trees are rooted in the same

location. Each tree is generated by a different CPU. Their findings show that the shared information between the different trees can be highly beneficial. It allows the trees to corporate as they explore unknown test spaces. This method increased speed and efficiency. The images and data within Chapter 3 show that the MA-RRF results are in sync with [48]. In this thesis, the shared database between all trees was used during the forest linking process. These MA-RRF multi-tree links encourage cooperation between the different trees. Here, the nodes from multiple trees cooperate to create a single path. These findings show that the shared information creates a cooperative MA-RRF planner that is faster than an independent multi-tree RRT system.

Many researchers have chosen to use GA as a UAV trajectory planner. Some works apply GA as a stand-alone path planner whereas others use it alongside other algorithms. Path planning for quadrotor UAV using GA is presented by [109]. Their test space had minimal obstacles and the final path was a straight line. Their findings show that GA was run for more than 500 generations in order to find short term path nodes. Similarly, a multiple coordinated agents' coevolution EA (MCACEA) is applied within [112] to generate paths for multiple UAVs. Their results also show that the path planner can require almost 500-1000 generations in order to fully optimize the multi-agent trajectories. Initial experimentation for this research project matches the conclusions of [109] and [112]. It showed that using GA as a stand-alone free space mapping and planning algorithm will require a longer processing time. The published work [111] implements a modified breeder GA (BGA). Here, BGA is combined with Bspline and VPF in order to generate UAV path nodes across mountainous terrains. Their long-term offline planner only required a population size of 100 and 50 generations to reach a good solution. This shows that a hybridized algorithm can be faster than a stand-alone GA path planner. The work that is presented in this thesis is in agreement with [111]. The combination of MA-RRF as the initial search algorithm and GA as an optimizer creates a faster path planner. The results within Chapter 3 also show that this combination facilitated the quick creation of thousands of hybridized paths across many generations.

The initial inspiration for this hybridized trajectory planner was the findings within [50]. The authors of this paper applied a two tree RRT sampling-based planner in order to generate path nodes across protein conformations. Their work showed that the many paths that were extracted from the RRT planner can be hybridized to create one superior path. This study emphasized that the quality of some path subsections may be better than the quality of an individual path. The quality of each subsection was measured through a cost matrix that prioritizes shorter pathways. Lastly, the path subsections are fused together to create a more optimal final trajectory. The results that were presented in Chapter 3-5 are in agreement with the conclusions of [50]. Here, the path subsections that were generated by the initial MA-RRF planner are hybridized to create more optimal child paths within GA.

This thesis also shows that MA-RRF can also be applied successfully within GA to repair the hybridized paths. This research project has expanded the limitations within [50] by generating tens of thousands of hybridized paths across all generations. It also applies a more refined many-objectives optimization algorithm to rank the path subsections. The final paths that are shown in Chapter 5 can be compared to the initial path population in Chapter 3. It is similar to the outcome of study [50]. The final hybridized paths are of higher quality and shorter than the initial MA-RRF paths.

One of the most important contribution of this work is the control system for the multi-agent quadrotors. Firstly, real life quadrotor UAVs must be able to track the designed hybridized trajectories. [143] shows that generating minimal snap trajectories allows the quadrotors to track the path nodes with minimum deviation error. The paths within this thesis are converted into minimal jerk trajectories using B-spline curves. The findings that are presented in Chapter 5 show that there is still a small amount of deviation error that is present across all generations. The error value is within close range to the worst case positional error that is obtained within [143]. Their experimental results show that the quadrotors are still capable of tracking complex manoeuvres with large accelerations as a team. Next, the simplified mathematical model for each agent is inspired by the full model within [140]. The author also presents a control system that defines the positional and rotational control structures. The simulation results within Chapter 3 are synonymous to those that are presented in [140]. The author places importance on using a controller that is capable of stabilizing a system by maintaining a zero value for all three of the Euler angles. The estimated values for each quadrotor within this research project show that the PD controller was able to fulfil this requirement.

Most publications on optimization algorithms will define the costs, limitations and objectives of the UAV's mission. The costs that are applied within these works are often specific to the type of UAV, number of agents, task and environment. [112] presented a trajectory planner for multiple UAVs that aims to minimize 11 objectives. Their work considers costs such as the path length, altitude, slope and curvature. It also takes into account the map limits, possible collisions, no fly zones and fuel consumption. The importance if each objective is defined through its predefined priority level. Similarly, [113] applied 7 different objectives within its real-time UAV path planning algorithm. These costs penalize longer paths that require more fuel. It also penalizes paths with nodes at higher altitudes or within danger zones. Lastly, paths with extreme curvatures and possible collision points are penalized as well. The objective functions that were presented in Chapter 4 are similar to those that were presented within [112-113]. These costs form the standard objectives that were applied within both the MA-Spread and MA-Formation missions. The findings in Chapter 5 are in agreement with [112-113]. The results show that these objectives are crucial to generating feasible trajectories. These costs encourage the

maintenance of paths that can be easily implemented within real-life multi-agent UAV missions.

Some multi-agent missions require the agents to fly independently whereas others demand that the agents fly in formation. The researchers of each mission will have different objectives to accomplish. The most important objective within most spatially spread missions is the gathering of information. This is achieved in [28] by implementing an Information-rich RRT (IRRT) motion planner for quadrotor UAVs. Their work aims to reduce environmental uncertainty. [28] finds a balance between minimizing the total cost of a mission whilst maximizing the information that is collected by the agents. This is accomplished by implementing a corporative multi-agent quadrotor system. Each agent takes into account the path and information content of other quadrotors as it designs its own information-rich trajectory. Their findings show that a noncooperative system collected more information at the expense of higher cost values. The results of the corporative multi-agent quadrotor UAV path planning system in thesis differ from [28]. The results of the MA-Spread mission in Chapter 5 show that a path planner can minimize the mission costs and reduce environmental uncertainty simultaneously. It also shows that the collection of redundant sensory data can be minimized as well. This way the end user will not have to degrade one objective for another within a corporative multi-agent system.

The second type of multiple-UAV system requires the agents to fly in formation. Study [142] tests three types of formation flights. The authors make a comparison between decentralized, leader-follower and virtual structure trajectory generation. Their findings show that the agents are capable of maintaining their formation shape well when a virtual structure is used as a guide. In this case, the agents are defined as a rigid body. The results of [142] state that applying a virtual structure is also the most computationally intensive method of formation planning. The dynamic formation planner within this thesis was defined in Chapter 4. The findings of this research project are in agreement with [142]. This dynamic formation planner applies a virtual structure and a decentralized control system. The system is more fault tolerant because each agent is independent. It is also more complex and requires a longer processing time. This research project is still successful because isn't highly affected by a longer processing time because it is designed to be an offline planner. It is important that a formation planner is capable of designing obstacle free virtual structures. The design of the high resolution adaptive formation planner within this thesis is inspired by [141]. The authors use VPF to spread a swarm of agents across predefined polygonal shapes. The agents are attracted to the centre of the virtual shape. They are also repulsed by their fellow neighbouring agents. This system results in the agents being well spread across the virtual formation structure. The images within Chapter 5 are in agreement with [141]. The results show that the quadrotor UAV are well spread

across the free spaces within each test environment. The agents are able to contract and expand along with their virtual structures.

The many objectives that were previously defined for the MA-Spread and MA-Formation mission will be used to rank the trajectories. The ranking process within a many objectives optimization problem can be highly challenging due to the large number of dominant solutions. The multi-agent trajectory sorting process was performed by the DRMOO algorithm as defined in Chapter 4. The DRMOO algorithm is built upon the concept of dimensionality reduction that was described in detail within [138] and [133]. Their findings showed that some objectives within the many-objective optimization problem can be labelled as redundant. These objectives are considered to be redundant to the ranking process if their removal doesn't affect the set of dominant solutions. Both studies have shown that the removal of these redundant objectives will increase the selection pressure towards a Pareto optimal solution set. The DRMOO algorithm also expands the usage of objective subsets as seen in [136]. The authors break down the many objectives into randomly selected objective subsets. Their work applies both the full and partial objective sets in rotation across generations. The authors show that combining both the full and partial objective sets can create a well-spread and well-converged final solution set.

The results that were presented in Chapter 5 show are similar to the findings of [138,133,136]. Firstly, dimensionality reduction was applied to create objective subsets within this thesis. The data that was shown in Chapter 5 shows that this process successfully increased the selection pressure within each generation. Secondly, the full and strategically created objective subsets were applied in rotation across generations. This study differs from [136] because the objective subsets were not created randomly. The findings within Chapter 5 show that implementing partial Pareto dominance ranking along with dimensionality reduction can minimize or maintain the values of all objectives. This is achieved without degrading any objective for the sake of another. The end user was presented with a diverse and well minimized collection of multi-agent trajectories at the end of the optimization process.

6.3. LIMITATIONS OF THIS THESIS

This project was designed with a few limitations in terms of research scope. The experiments and findings within this study are restricted to simulation only. The path planning system is designed to mimic real-life multi-agent quadrotor UAV flights. The test environments are also assumed to be well known and created to mimic the common quadrotor test spaces. The only environment that isn't included is over and underwater. These partly known environments can be generated by the end user through initial UAV

sensory data or with an online map database. This system will allow the user to simulate the flight process with a certain level of accuracy before performing hardware testing.

The design and analysis within this study is focused on long term path planning with static obstacle avoidance. Many works are based on short term online path planning with dynamic obstacle avoidance. This research aims to fill the need for long term offline path planning across areas that have minimal dynamic obstacles. It is designed for areas such as evacuated disaster sites. It can be used for environmental crisis, medical supplies transportation or weaponry identification. The end user has the option of implementing an ultrasonic sensor if basic dynamic obstacle avoidance is necessary.

This study is also primarily concerned with implementing algorithms that are fast and minimal error. This can differ from projects that require highly accurate models of their UAV and its control system. This work only addresses the estimation of the agent's flight path and states. It applies a simplified version of the mathematical model that has been proven by other researchers to be sufficient for estimating the quadrotor's movements. Similarly, a noncomplex PD controller is applied despite having a small amount of deviation error. This is because speed is extremely important when tens of thousands of paths are being evaluated for many agents. The states estimation for all agents must be done simultaneously through a parallel run multi-agent control system.

Similarly, highly complex mathematical models for the objective functions aren't implemented within this research. These cost functions are used to provide an estimation of the pros and cons for each trajectory. It will not produce an accurate real-life representation of the cost values. It is enough to offer the end user insight into which trajectory within the entire collection is best for their mission. The estimated objectives values are used within the dimensionality reduced many-optimization process to rank the trajectories. The DRMOO algorithm is not concerned with making sure each objective is conflicting with all the others within an objective subset. It only aims to continue to check and improve the chances of an objective residing within an objective set that contains conflicting costs. The findings show that this is sufficient to improve the selection pressure within the optimization process.

This research has deliberately avoided fully defining the Pareto front during the many-objectives optimization process. As previously defined, the primary focus is set on maintaining or minimizing the values of an objective without the degradation of the others. It brings the collection closer to the Pareto optimal front as opposed to defining the exact Pareto optimal solutions. This work also concentrates on maintaining a well minimized and diverse population of trajectories. It aims to maintain or improve the level of diversity between the initial MA-RRF to the final GA population. Thus, it is not concerned with

accurately representing a well spread Pareto front. This research is restricted to providing the end user with a large collection of well minimized and diverse multi-agent trajectories.

The final constraint in terms of study scope is the termination point of the DRMOO algorithm. It is important to note that the final population is determined through human analysis of visual imagery and the level of objective minimization. The visual study easily shows the diversity of the path population. It simplistically defines is the paths are spread across different areas. The data shows if the population is well minimized in comparison to the initial MA-RRF population. This research intentionally tries to find a balance in trajectory optimization and diversity. In this case, the final population may not hold the minimal value for all objectives. Here, it must be stressed that the end user has tens of thousands of path choices across 100 generations to choose from. It is possible for the user to pick a different termination point.

6.4. ISSUES WITHIN THE RESEARCH PROCESS

The experimental process of any research project can be challenging. In most cases, there are often issues that occur during the simulation and testing process. The first challenge within this project is to design test environments that are suited to the multi-agent quadrotor missions. Each environment is complex and it can be difficult for the viewer to see clearly that the path planner is capable of generating nodes that do not collide with the obstacles. Still, it was important that these simulated test spaces contain sufficient detail in order to mimic real life environments.

A balance between creating easy to interpret imagery, real-life accuracy and processing time was necessary. This was done by limiting the number of clutter within each room in the indoor space. Similarly, the size of the mountainous terrain was reduced in order to preserve the gradual height changes within the peaks and lows. Then, it was a challenge to define the size of the safety boundaries around each obstacle. The safety zones of the mountainous terrain had to accurately represent the changes in height across the terrain. Here, the heights of the safety zones are based on the maximum height of the mountain peaks that are within it. It was especially challenging to determine the boundary size within the cityscape and indoor environment. A larger sized safety zone will minimize possible obstacle collisions but reduce size of the already narrow passages. Similarly, smaller safety zones will allow more agents to fly across these narrow passages but increase the chances of collisions. The boundary size within these two environments was determined through experimentation and by considering the average real-life sizes of quadrotors.

There are some issues that must be tackled when creating a standardized platform for two different applications. The only difference between the two missions is the addition of a dynamic formation planner. In this research, the same path planning and optimizing algorithms is applied within the multi-agent quadrotor spatially spread and formation flight mission. Firstly, the MA-RRF planner was designed to take advantage of a multi-agent system by generating multiple trees. This process generates feasible paths for many agents simultaneously within the MA-SPREAD application. On the other hand, the MA-FORMATION mission only requires one collection of formation reference paths. Thus, there is only one MA-RRF tree. The same algorithm is usable by implementing virtual agents across the test space. This method quickly generates a diverse collection of formation reference paths. Similarly, the same GA and DRMOO algorithm is modified to be applicable within both missions. There is also a standardized set of objectives for both tasks. Here, the end user has the option of introducing four more costs that are specific to either the spread or formation mission.

Another challenging aspect of any study is the identification of unknown variables. All of the subsections within this research contain variables or operators that will influence the final results. The MA-RRF path planner requires a sampling node, forest link and near goal node range. The values of these variables are set based on the level of diversity of the resulting path population. Similarly, GA has multiple operators within its structure. The type of operators within the GA can improve or destroy optimal paths. These operators are set based on the maintenance of optimal solutions within the resulting population. Next, determining the percentage for the path similarity threshold within the MA-RRF and GA can be tough. The algorithm will require a much longer run time if a high level of diversity is required. Many optimal paths will be removed from the population. A balance was found by implementing a percentage that isn't too strict but encourages path diversity. Lastly, the final variable that needs to be determined is the termination point for the MA-RRF and GA algorithm. In both cases, the termination point is a constant value that has been determined through experimentation. The number of iterations allows the MA-RRF planner to generate a large collection of initial paths and repair child paths within GA.

All of the subsections within the hybridized path planner will require a well-organized shared database. This open database provides the end user with additional information regarding the trade-offs of each trajectory. It contains data from the MA-RRF planner, GA and DRMOO. These values will require a huge amount of storage and must be easily extractable. The testing process showed that the manner in which the data is stored can affect the speed of an algorithm. Initially, the path extraction and filtration process required a longer completion time. The MA-RRF planner contains multiple trees and forest linkages between them. It was important to store the connections between the branches systematically so that the path subsections can be easily obtained. The estimated states for

each agent have to be easily extractable by the DRMOO subsection. Similarly, the objective values and ranking for each trajectory must be arranged in a manner that isn't difficult for the end user to understand. The results show that optimal trajectory planner is capable of running smoothly with an organized database.

6.5. IMPLICATIONS OF RESULTS

The results and findings within this research project provide support for the argument that many-objectives optimization can be applied towards multi-agent UAV problems. Current commercially available processing systems are not capable of performing many objectives path optimization for long term planning on board an UAV. A ground base station will also be incapable of completing the optimization process quickly. The generation, estimation and ranking of many trajectories cannot be done in real time. Still, this study shows that it is possible to perform long term path planning for many agents offline. This can be done within a one-day time frame and only requires a basic multi-thread processing system. It can be useful for multi-agent UAV missions that require some level of accuracy and information before execution. This planner is suitable for war and disaster zones that contain minimal living beings. It is also suited towards unexplored terrains.

The optimal trajectory planner also offers evidence that a standardized system can be applied towards both spatially spread and formation flight missions. Many works are focused on either cooperative or noncooperative flight tasks. This study proves that it is possible to create a standardized platform for both options. This multi-agent UAV path planner only requires the addition of a dynamic formation planner when switching between missions. Next, the results provide evidence that it isn't always necessary to implement a weighted priority or token system that can cause bias within the optimization process. Many studies choose to focus on a selected number of objective functions when creating a path planner for UAVs. This research project shows that it is possible to implement any number of objectives within the trajectory optimization process. It shows that the many objectives can be minimized simultaneously without the degradation of the others.

The work that is presented within this thesis proves that a fast-parallel run system can be implemented with a multi-agent quadrotor team. All of the subsections are created to perform faster within a multi-thread processing system. The trees within the MA-RRF planner can be generated in parallel. Similarly, the multi-agent child paths that are generated by the GA can be designed in parallel. Lastly, the closed-loop control system is designed to output the predicted states of all agents simultaneously. This is also achieved by running the control system of each agent in parallel.

This research also offers evidence that a path planning algorithm is capable of offering a large collection of diverse and optimized trajectories for many agents simultaneously. This is different from popular studies that aim to offer a singular best path. Lastly, the results suggest that it is possible to present the end user with a well-organized database that is filled with easy to understand information. Many drone users will welcome the addition of easy to understand imagery and knowledge regarding the designed multi-agent trajectories.

6.6. FUTURE RECOMMENDATIONS

Today, path planning for unmanned vehicles continue to be a popular research topic. This is because commercially available drones such as the quadrotor are often used by both professionals and hobbyists. Future researchers can either choose to expand or reduce the scope of this project. This research project is a fully simulation based study. Thus, the future recommendations of the project will be focused on possible simulation based topics.

One way of expanding this study's scope is by implementing a much larger number of objectives, costs or limitations within the optimization process. As previously defined, most decisions that humans face in real life are based on many costs. The introduction of a larger collection of objectives will allow the end user more flexibility when applying the system towards their desired mission. It will allow the end user to attach the cost functions that are suited to their current multi-agent UAV task. Introducing more objectives will further standardize the optimal path planner. Thus, it will be applicable within a large variety of real life tasks.

Another method of pushing the limitations of this project is by applying the algorithm with a swarm of quadrotor UAVs. This project applied the path planner with 8 quadrotors. The optimal trajectory planner can be expanded to include a larger number of agents. This will require a fast swarm based control system that is capable of estimating the states of each agent. These experiments will test if the completion time of the path planner increases or decreases in speed with a large number of agents. The most challenging subsection will be the dynamic formation planner. A swarm of agents will require a lot of coordination in terms of formation structure and maintenance.

New researchers can further spread the scope of this project by performing long term path planning across much larger distances. A much larger search area will contain a variety of environments as opposed to just one. An urban space can contain both office and residential areas. Thus, the algorithm will have to generate paths across a cityscape and multiple highly cluttered indoor environments. Current researchers can also choose to minimize the scope of this project when proposing a new research topic. They can choose

to apply the optimal path planner for specific tasks. The planner can be implemented within missions such as payload transportation, target tracking, weaponry identification or agricultural and wildlife imaging.

Another manner that this project can be implemented at a smaller scale is by applying it within a smaller test space. This converts this long-term planner into a short term one. In this case, real-time node generation and path replanning can be performed in real time. Dynamic obstacle avoidance can also be introduced. The optimization process within the algorithm will have to be simplified. The path planner will also require a fast-multi-thread processing system in order to perform in real-time.

All of these possible future topics can provide new findings and contributions to the field of robotics. It will continue to provide the end user with a large collection of feasible solutions. These topics will also offer the end user more information and assistance in regard to which solution is the best for their mission.

6.7 CONTRIBUTIONS TO RESEARCH

The first target of this research was to construct and map three-dimensional test environments that illustrate real-life flight challenges. The objective of designing real world locations for the purpose of standardizing the trajectory planner is well achieved. Three defining environments were designed based on prior popular research projects across the world. These various environments were designed to test the robustness and adaptability of the hybridized algorithm in the face of a variety of different challenges and constraints. The three environments applied here are the high-rise cityscape, highly cluttered indoor environment and mountainous terrain. All three are locations that are available across the world and are locations where quadrotors are most commonly applied. The safety boundaries within each test space were also successfully integrated into these environments.

Next, the path planner aimed to generate highly diverse collision free paths for the initial population of the optimization process. The diversity of these initial paths is paramount to the effective application of the many-objective optimization algorithm. This study has successfully achieved this target of based on the multi-agent paths that were extracted by the MA-RRF algorithm. The MA-RRF algorithm achieved high levels of free space exploration with zero obstacle collisions. Results show that the multi-agent RRF system is speedy and has an advantage over the basic RRT algorithm. It used the multi-agent system to its advantage by generating a multi-tree forest. Here, the cost function for possible forest links is easily modifiable to suit the end user. There were challenges within the more constricted environments. In these cases, extracting diverse paths was difficult

due to the presence of narrow passages and large amounts of obstacles. The MA-RRF was still capable of free space mapping and path planning despite these limitations. Thus, the MA-RRF design can be reapplied as a standalone platform when speed is required. It can be applied for critical situations such as real-time collision avoidance or path replanning.

Another target that must be achieved by the optimal trajectory planner is the creation of a unique population of multi-agent paths for each generation. The initial suboptimal trajectories that were developed by the MA-RRF algorithm were applied as the input for the optimization process. This process reduced the amount of free space that needs to be searched by GA. It also allowed GA to execute more refined multi-agent path planning. The initial MA-RRF trajectories were meshed to create new paths through the GA's crossover and mutation process. MA-RRF was also applied for the path repair of child trajectories after the crossover and mutation process within GA. Four additional post processing GA operators were well implemented in order improve the survivability of these child paths.

It is important that the paths that are created must be converted into time-based smooth trajectories that are suitable for multi-agent quadrotors. The paths from the GA are successfully converted from a collection of nodes to time-based trajectories through adaptive minimal jerk splines. One important design factor in the spline implementation was its adaptability across extreme angles between two nodes. Here, the addition of extra nodes enhances the smoothness of the turn. It also stopped the spline from crossing across an obstacle when two nodes are too far apart. The minimization of sudden jerks proves to create smoother transitions between waypoints. This ensures continuity for the second order derivatives of the quadrotor's roll and pitch angle. Despite the achievement of smooth design, the quadrotor was chosen based on its capability for aggressive manoeuvring. Thus, the end user always has the option of choosing between smoother and more aggressive trajectories.

Then, the study intends to model a noncomplex parallel run closed-loop multi-agent quadrotor control system. The objective of successfully designing a quadrotor control system for speedy tests of feasible paths is complete. The design of the control system mimics the flight trajectories of the multi-agent quadrotors. It contained two subsections for each agent. The first stage holds the PD control system whereas the latter holds the mathematical model of the quadrotor. The desired trajectory nodes for all agents are applied simultaneously within the control system. This generates the estimated positional and rotational states of each agent. These states were then successfully used to predict the values of the many objective functions.

This research also aimed to identify and standardize the various applications of multi-agent quadrotors. Thus, the optimization algorithm is applicable towards any variation of real world multi-agent cooperative tasks. The algorithm successfully defined and optimized paths for both the multi-agent spread and formation flight mission. The MA-SPREAD was well implemented to create a common trajectory planner for missions such as search and rescue, surveillance, reconnaissance, terrain mapping, multi-target tracking, environmental monitoring, forming ad hoc wireless networks, wildlife and atmospheric research and disaster relief. Whereas, MA-FORMATION was successfully designed for payload transportation, military missions such as security patrols, search and rescue at hazardous disaster sites, cooperative sensory angular coverage and aerial flights whilst maintaining precise patterns.

Another goal of this work is to mathematically define a set of multi-agent quadrotor mission-based objectives. A standardized definition of the quadrotors' physical constraints and mission limitations were represented by a collection of objectives functions. These objective functions are applicable to any environment, any number or variation of quadrotor shape, size or mass as well as other options of network range, sensors, control method and measurement unit. The optimization of multi-agent quadrotors is performed as a team for both missions. The MA-SPREAD mission optimized the combination of multi-agent trajectories. The objectives from the MA-SPREAD mission are dependent on the team's terrain exploration. On the other hand, the MA-FORMATION mission optimized the formation reference trajectory at every generation. The objective functions for formation flight are dependent on the formation shapes that are planned by the dynamic formation planner. As previously discussed, any additions of new functions within the many-objective optimization algorithms is as simple as adding it into the objective set with no modifications to the algorithm.

This study also placed a lot of importance on developing a well optimized yet diverse final set of trajectories for end users. The values from the many objectives were used to sort the multi-agent trajectories based on their level of optimality and diversity. Here, DRMOO ranking algorithm was defined in detail. This algorithm combines both partial dimensionality sorting with full high dimensionality optimization. In this research, we applied dimensionality reduction in order to increase selection pressure without the absolute removal of any objectives. The objective subsets were not created randomly. It was performed by evaluating the level of conflict between objective pairs within each subset. This process minimized the chances of full elimination of an objective function and leaves room for possible error. Here, the application of adaptive niching contributed another level of diversity maintenance. The algorithm is designed to focus on the minimization of all objectives as opposed to mapping the Pareto front with full accuracy.

Thus, the final set of trajectories as well as its data is an approximate of the Pareto optimal solutions.

Lastly, this work intends on providing well organized and easily understandable information regarding the trade-offs of each solution. The results showed that the algorithm performs successfully in finding a diverse set of optimal solutions within each environment. The end user is supplied with high resolution visual imagery and well-organized data. The additional knowledge will assist the end user in making a final choice. In the MA-SPREAD mission, the end user is able to compare the sensory data overlap, uncertain terrain coverage and network connectivity when choosing the best option. In the MA-FORMATION application, the end user is provided with a variation of solutions where formation speed, rigidness or simplicity can be used as the final decision factor.

REFERENCES

[1] G. Apostolo, "The Illustrated Encyclopedia of Helicopters", 1984.

[2] J. W. R. Taylor, "Jane's Book of Remotely Piloted Vehicles", Collier Books, 1977.

[3] Parrot Designs, "A. R. Drone 2.0", Internet: http://www.parrot.com/usa/products/ardrone-2/, 2016.

[4] Gizmag, "Review: Parrot's Rolling Spider Minidrone", http://www.gizmag.com/parrot-rolling-spider-minidrone-review/33840/, 2016.

[5] Parrot SA, "Parrot Minidrone Rolling Spider", Internet: http://www.parrot.com/usa/products/rolling-spider, 2015.

[6] Draganfly Innovations Inc," Lil' Draganflyer Plus Mini Quadrotor", Internet: http://www.draganfly.com/industrial/products.php , 2015.

[7] W.Mielniczek, " 'B' the flying car", Internet: https://www.kickstarter.com/projects/2017062404/b-go-beyond, 2015.

[8] Latscha,S., Kofron,M., Stroffolino,A., Davis,L., Merritt,G., Piccoli,M., Yim,M., "Design of a Hybrid Exploration Robot for Air and Land Deployment (H.E.R.A.L.D) for urban search and rescue applications", International Conference on Intelligent Robots and Systems (IROS 2014), pp. 1868 - 1873, Chicago, IL, September 2014.

[9] C. E. Doyle, J. J. Bird, T. A. Isom, J. C. Kallman, D. F. Bareiss, D. J. Dunlop, R. J. King, J. J. Abbott, and M. A. Minor, "An Avian-Inspired Passive Mechanism for Quadrotor Perching," IEEE/ASME Transactions on Mechatronics, vol 18(2), pp. 506-517, 2013.

[10] D. Schneider, " Helicopters Go Electric", IEEE Spectrum, Internet: http://spectrum.ieee.org/aerospace/aviation/ helicopters- go-electric, December 2011.

[11] Malloy Aeronautics (MA), " The Hoverbike helicopter ", Internet: http://www.hoverbike.com/MA/product/ hoverbike-helicopter, 2014.

[12] M. M. Maia, P. Soni, F. J. Diez-Garias, "Demonstration of an Aerial and Submersible Vehicle Capable of Flight and Underwater Navigation with Seamless Air-Water Transition", Rutgers University, 2015.

[13] Gizmag, "Pars life-saving flying robot is now a reality", Internet: http://www.gizmag.com/pars-life-saving-flying-robot/29831/, 2013.

[14] Unmanned Systems Technology, "microdrones Alps Crossing", Internet: http://www.unmannedsystems technology.com /2013/06/video-microdrone-uas-completes-first-flight-across-the-alps/microdrones-md4-1000-alps-crossing/, June 2013.

[15] J. Q. Cui , S. Lai, X. Dong, P. Liu, B.M.Chen, T.H.Lee, "Autonomous navigation of UAV in forest", International Conference on Unmanned Aircraft Systems (ICUAS), pp. 726 – 733, Orlando, May 2014.

[16] R. Deits, R. Tedrake, “Efficient Mixed-Integer Planning for UAVs in Cluttered Environments”, IEEE International Conference on Robotics and Automation (ICRA), May 2015.

[17] N. Tran, C. Nguyen, D. Vu , T. Hoai, “Embedded-oriented techniques for 2D shortest trajectory planning to avoid restricted airspaces”, International Symposium on Communications and Information Technologies (ISCIT), pp. 238 - 242, Incheon, September 2014.

[18] A. Giusti, J. Guzzi, D.C. Cireşan, F. He, J. P. Rodríguez, F. Fontana, M. Faessler, C. Forster, J. Schmidhuber, G. Di Caro, D. Scaramuzza, L. M. Gambardella, “A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots”, IEEE Robotics and Automation Letters, vol.1, pp. 661 – 667, July 2016.

[19] B. Lam, “Incredible Close-Up Drone Video of an Erupting Volcano in Iceland”, Internet: http://www.wired.com/2014/10/drone-video-iceland-eruption-bardarbunga-volcano, October 2014.

[20] K. Nagatani, “Robotic Observations in a volcanos using UAV and UGV in Mt. Asama”, Internet:http://www.astro.mech.tohoku.ac.jp/~keiji/Research/research.html, October 2014.

[21] E. Ackerman “Robocopters Haul Tons of Stuff in Afghanistan, Return Home Victorious”, Internet: http://spectrum.ieee.org/automaton/robotics/military-robots/kmax-robocopters-haul-tons-of-stuff-afghanistan, July 2014.

[22] I. Lamcja, “Canada’s police forces take to the sky with drones”, Internet: http://metronews.ca/news/canada/1314670/canadas-police-forces-taking-to-the-sky-with-drones, March 2015.

[23] N. Mathew, S.L. Smith, S.L. Waslander, “Planning Paths for Package Delivery in Heterogeneous Multirobot Teams”, IEEE Transactions on Automation Science and Engineering, vol.12, pp. 1298 - 1308, October 2015.

[24] Parrot Store, "Indoor Hull for AR. Drone 2.0 Elite Edition - Snow", Internet: https://store. parrot.com/uk/accessoires-ar-drone-20/36-indoor-hull-for-ardrone-20-elite-edition-snow-3520410019081.html, 2015.

[25] M. Pivtoraiko, D. Mellinger and V. Kumar, “Incremental Micro-UAV Motion Replanning for Exploring Unknown Environments”, IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, May 2013.

[26] S. Tang, V. Kumar, “Mixed Integer Quadratic Program trajectory generation for a quadrotor with a cable-suspended payload”, IEEE International Conference on Robotics and Automation (ICRA), June 2015.

[27] D. J. Webb, J.V.D Berg, “Kinodynamic RRT*: Optimal Motion Planning for Systems with Linear Differential Constraints”, May 2012.

[28] D.Levine, B.Luders, J.P.How, "Information - Theoretic Motion Planning for Constrained Sensor Networks," Massachusetts Institute of Technology, July 2012.

[29] European Space Agency, “Smartphone App Turns Home Drone into Spacecraft”, Internet:http://www.esa.int/Our_Activities/Space_Engineering_Technology/Smartphone_app_turns_home_drone_into_spacecraft, March 2013.

[30] CNN, “When is my personal drone landing?”, Internet: http://edition.cnn.com/2013/12/17/opinion/calo-drones-apps/, December 2013.

[31] LaValle, S. M., Planning Algorithms, Cambridge University Press, Cambridge, U.K., 2006.

[32] M. Nieuwenhuisen, S. Behnke, “3D Planning and Trajectory Optimization for Real-time Generation of Smooth MAV Trajectories”, European Conference on Mobile Robots (ECMR), Lincoln, UK, September 2015.

[33] V.R. Desaraju and N. Michael, “Hierarchical Adaptive Planning in Environments with Uncertain,Spatially-Varying Disturbance Forces”, 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 5171 – 5176, Hong Kong, June 2014.

[34] O. Saif, I. Fantoni, A. Zavala-Rio, “Real-time Flocking of multiple-quadrotor system of systems”, 10th IEEE System of Systems Engineering Conference (SoSE), San Antonio, TX, United States, May 2015.

[35] H. A. F. Almurib, P. T. Nathan, and T. N. Kumar, “Control and Path Planning of Quadrotor Aerial Vehicles for Search and Rescue”, SICE Annual Conference 2011, Waseda University, Tokyo, Japan, September 2011.

[36] G. M. Hoffmann, S. L. Waslander, “Quadrotor Helicopter Trajectory Tracking Control”, AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii, August 2008.

[37] Mac Schwager, Jean-Jacques Slotine, Daniela Rus, “Unifying Geometric, Probabilistic, and Potential Field Approaches to Multi-robot Coverage Control”, Robotics Research, vol.70, pp 21-38, 2011.

[38] A. Pierson, A. Ataei, I. Ch. Paschalidis, and M. Schwager ,“Cooperative Multi-Quadrotor Pursuit of an Evader in an Environment with No-Fly Zones”, Stanford University, April 2015.

[39] L. Tapia, A. Faust, N. Malone, H. Chiang, K. Manavi, “Motions for Complex and High-Dimensional Robots - Planning with Model Uncertainty”, Center for Advanced Research Computing, University of Mexico, 2014.

[40] P. M. Bouffard, S. L. Waslander, “A Hybrid Randomized/nonlinear Programming Technique for Small Aerial Vehicle Trajectory Planning in 3D”, IROS 3rd Workshop: Planning, Perception and Navigation for Intelligent Vehicles, 2009.

[41] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, J. P. How, "Motion Planning in Complex Environments using Closed-loop Prediction," AIAA Guidance, Navigation, and Control Conference, Honolulu, August 2008.

[42] C.Richter, A.Bry, and N.Roy, "Polynomial Trajectory Planning for Quadrotor Flight, "International Symposium of Robotics Research, 2013.

[43] D.Levine, B.Luders, J.P.How, "Information - Theoretic Motion Planning for Constrained Sensor Networks," Massachusetts Institute of Technology, July 2012.
[44] Desaraju, V.R., How, J.P., "Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees", IEEE International Conference on Robotics and Automation, pp. 4956–4961, 2011.
[45] Kothari, M., Postlethwaite, I., Gu, D.-W., "Multi-UAV path planning in obstacle rich environments using Rapidly-exploring Random Trees", IEEE Conference on Decision and Control, pp. 3069 - 3074, December 2009.
[46] A. A. Neto, D. Macharet, L. Chaimowicz, M. Campos, "Path planning with Multiple Rapidly-exploring Random Trees for teams of robots", 16th International Conference on Advanced Robotics (ICAR), pp. 1 - 6, November 2013.
[47] S.K.Nath, S.Thomas, C.Ekenna, N.M.Amato, "A Multi-Directional Rapidly-Exploring Random Graph (mRRG) for Protein Folding, "ACM Conference on Bioinformatics, Computational Biology and Biomedicine,pp.44-51,October 2012.
[48] M.Otte, N.Correll, "C-FOREST: Parallel Shortest-Path Planning with Super Linear Speedup,"MIT, 2013.
[49] D.Devaurs, M. Vaisset, T. Siméon, J. Cortés, "A multi-tree approach to compute transition paths on energy landscapes,"Workshop on Artificial Intelligence and Robotics Methods in Computational Biology, AAAI '13, July, 2013.
[50] A. Enosh, B. Raveh, O.Furman-Schueler, D. Halperin, N. Ben-Tal, " Generation, comparison, and merging of pathways between protein conformations: gating in K-channels,"Biophysical Journal, vol. 95, pp. 3850-3860, October 2008.
[51] R. O. Saber, "A Unified Analytical Look at Reynolds Flocking Rules", American Control Conference 2004, Boston, MA, Spetember 2003.
[52] R. Abbas, Q. Wu, "Tracking Formation Control for Multiple Quadrotors Based on Fuzzy Logic Controller and Least Square Oriented by Genetic Algorithm", The Open Automation and Control Systems Journal, vol.7, pp.842-850, 2015.
[53] Z. Hou, I. Fantoni, A. Zavala-Rio, "Modeling and Decentralized Control for the Multiple UAVs Formation based on Lyapunov design and redesign", 2nd IFAC Workshop on Research, Education and Development of Unmanned Aerial Systems, Vol.46, Issue.30, pp. 337-344, 2013.
[54] Z. Hou, I. Fantoni, "Leader-Follower Formation Saturated Control for Multiple Quadrotors with Switching Topology", 2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), pp. 8-14, Cancun, November 2015.
[55] Z. Hou, I. Fantoni, "Distributed leader-follower formation control for multiple quadrotors with weighted topology", 10th System of Systems Engineering Conference (SoSE), pp. 256 – 261, San Antonio, TX, May 2015.
[56] B. Shirani, M. Najafi, I. Izadi, "Cooperative Load Transport Using Multiple Quadrotors", 2nd Iranian Conference on Avionics System, February 2015.

[57] V. Roldão, R. Cunha, D. Cabecinhas, C. Silvestre, P. Oliveira, “A novel leader-following strategy applied to formations of quadrotors”, 2013 European Control Conference (ECC), Zürich, Switzerland, July 2013.

[58] G.P. Pereira, R. Cunha, D. Cabecinhas, C. Silvestre and P. Oliveira," Three dimensional trajectory planner for real time leader following," IEEE Conference on Robotics & Automation, pp. 6561 - 6566, June 2014.

[59] N. H. M. Li, H. H. T. Liu, “Formation UAV Flight Control using Virtual Structure and Motion Synchronization”, 2008 American Control Conference, pp. 1782 – 1787, Seattle, Washington, USA, June 2008.

[60] Abbas Chamseddine, Youmin Zhang ; Camille Alain Rabbath, “Trajectory Planning and Re-planning for Fault Tolerant Formation Flight Control of Quadrotor Unmanned Aerial Vehicles”, 2012 American Control Conference (ACC), pp. 3291 – 3296, Montreal, QC, June 2012.

[61] D. Zhou and M. Schwager, “Virtual Rigid Bodies for Agile Coordination of Quadrotor Swarms and Human-Swarm Teleoperation”, Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, August 2015.

[62] A. A. A. Rizqi, A. I. Cahyadi, T. B. Adji, “Path planning and formation control via potential function for UAV Quadrotor”, 2014 International Conference on Advanced Robotics and Intelligent Systems (ARIS), pp. 165 – 170, Taipei, June 2014.

[63] Rabah Abbas, Qinghe Wu, “Formation Tracking for Multiple Quadrotor based on Sliding Mode and Fixed Communication Topology”, 2013 Fifth International Conference on Intelligent Human-Machine Systems and Cybernetics, pp. 233 - 238, Hangzhou, August 2013.

[64] B. Yu, X. Dong, Z. Shi, Y. Zhong, “Formation control for quadrotor swarm systems: Algorithms and experiments”, 2013 32nd Chinese Control Conference (CCC), pp. 7099 – 7104, Xi'an, July 2013.

[65] Y. Kuriki, T. Namerikawa, “Formation Control of UAVs with a Fourth-Order Flight Dynamics”, SICE Journal of Control, Measurement, and System Integration, Vol. 7, No. 2, pp. 074–081, March 2014.

[66] Y. Kuriki, T. Namerikawa, “Consensus-based Cooperative Formation Control with Collision Avoidance for a Multi-UAV System”, 2014 American Control Conference, pp. 2077 – 2082, Portland, OR, June 2014.

[67] A. Chamseddine, Y. Zhang, C. A. Rabbath, “Trajectory Planning and Re-planning for Fault Tolerant Formation Flight Control of Quadrotor Unmanned Aerial Vehicles”, 2012 American Control Conference, Montréal, Canada, June 2012.

[68] D. Zhou, M. Schwager, “Virtual Rigid Bodies for Agile Coordination of Quadrotor Swarms and Human-Swarm Teleoperation”, Stanford University, October 2014.

[69] R. He, A. Bachrach, N. Roy, “Efficient Planning under Uncertainty for a Target-Tracking Micro-Aerial Vehicle”, 2010 IEEE International Conference on Robotics

and Automation Anchorage Convention District, Anchorage, Alaska, USA, May 2010.

[70] A. Kushleyev, D. Mellinger, V. Kumar, “Towards A Swarm of Agile Micro Quadrotors”, MIT Press, Edition 1, pp. 504, 2013.

[71] T. Lee, K. Sreenath, V. Kumar, “Geometric Control of Cooperating Multiple Quadrotor UAVs with a Suspended Payload”, IEEE Conference on Decision and Control, Italy, December 2013.

[72] A.Faust, I.Palunko, P.Cruz, R.Fierro, L.Tapia, "Learning swing-free trajectories for UAVs with a suspended load", IEEE International Conference on Robotics and Automation (ICRA), 2013.

[73] F. Wang, K. Wang , S. Lai , S. K. Phang , Chen, B.M. , Lee, T.H, " An efficient UAV navigation solution for confined but partially known indoor environments", 11th IEEE International Conference on Control & Automation (ICCA), pp. 1351 - 1356, June 2014.

[74] Barros dos Santos, S.R., Nascimento Junior, C.L., Givigi, S.N., “Planning and learning for cooperative construction task with quadrotors ", 8th Annual IEEE Systems Conference, pp. 57 - 64, April 2014.

[75] H. Huang, G. M. Hoffmann, S. L. Waslander, C. J. Tomlin, “Aerodynamics and Control of Autonomous Quadrotor Helicopters in Aggressive Maneuvering”, IEEE International Conference on Robotics and Automation, ICRA '09, pp. 3277 – 3282, Kobe, May 2009.

[76] C. Richter, A. Bry, N. Roy, “Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments”, Robotics Research, pp.649-666, January 2016.

[77] A. Symington, R. D. Nardi, S. Julier, S. Hailes, “Simulating Quadrotor UAVs in Outdoor Scenarios”, International Conference on Intelligent Robots and Systems (IROS 2014), Chicago, September 2014.

[78] Tomic,T., Schmid,K., Lutz,P., Domel,A., Kassecker, M., Mair, E., Grixa, I.L., Ruess, F., Suppa, M., Burschka, D.," Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue ", IEEE Robotics & Automation Magazine, vol.19, pp. 46 - 56, August 2012.

[79] Guerrero, J.A., Escareno, J.A., Bestaoui, Y., " Quad-rotor MAV trajectory planning in wind fields ", IEEE International Conference on Robotics and Automation (ICRA), pp. 778 - 783, May 2013.

[80] G. Hoffmann, S. Waslander, C. Tomlin. "Quadrotor Helicopter Trajectory Tracking Control", AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii, August 2008.

[81] A. Chamseddine, Zhang Youmin, C. A. Rabbath, C. Join, D. Theilliol , "Flatness-Based Trajectory Planning/ Replanning for a Quadrotor Unmanned Aerial Vehicle,"

IEEE Transactions on Aerospace and Electronic Systems, vol.48, pp. 2832 - 2848, October 2012.

[82] S.Waharte, N.Trigoni, S.J. Julier, “Supporting Search and Rescue Operations with UAVs ", International Conference on Emerging Security Technologies (EST), pp. 142 - 147, September 2010.

[83] J. Leonard, A. Savvaris, A. Tsourdos, “Energy Management in Swarm of Unmanned Aerial Vehicles”, 2013 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 124 – 133, Atlanta, GA, May 2013.

[84] M. Schwager, N. Michael, V. Kumar, and D. Rus, “Time Scales and Stability in Networked Multi-Robot Systems”, IEEE International Conference on Robotics and Automation, Shanghai, China, May 2011.

[85] J. L. Sanchez-Lopez, J. Pestana, P. D. L. Puente, R. Suarez-Fernandez, P. Campoy, “A System for the Design and Development of Vision-based Multi-robot Quadrotor Swarms”, International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, May 2014.

[86] J. Alonso-Mora, M. Schoch, A. Breitenmoser, R. Siegwart and P. Beardsley, “Object and Animation Display with Multiple Aerial Vehicles”, International Conference on Intelligent Robots and Systems, Portugal, October 2012.

[87] J. Leonard, A. Savvaris, A. Tsourdos, “Towards a Fully Autonomous Swarm of Unmanned Aerial Vehicles”, UKACC International Conference on Control 2012, Cardiff, UK, September 2012.

[88] L. Cantelli, M. Mangiameli, C.D. Melita, G. Muscato, “UAV/UGV cooperation for surveying operations in humanitarian demining”,

[89] Soltero, D.E., Schwager, M., Rus, D., "Generating informative paths for persistent sensing in unknown environments", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2172 - 2179, October 2012.

[90] R. He, Prentice, S. ; Roy, N., "Planning in information space for a quadrotor helicopter in a GPS-denied environment", IEEE International Conference on Robotics and Automation, pp. 1814 - 1820, May 2008.

[91] U. Pilz, A.P. Popov, H. Werner," Robust controller design for formation flight of quad-rotor helicopters", IEEE Conference on Decision and Control, pp. 8322 - 8327, Shanghai, December 2009.

[92] A.Kushleyev, D. Mellinger, C. Powers, V. Kumar," Towards a swarm of agile micro quadrotors," Robotics:Science and Systems VIII, MIT Press, 2013.

[93] M.Turpin, N.Michael, V.Kumar, "Trajectory design and control for aggressive formation flight with quadrotors," Autonomous Robots, volume 33, pp. 143-156, August 2012.

[94] R.W. Beard, T.W. McLain, D.B. Nelson, D. Kingston, D. Johanson, "Decentralized Cooperative Aerial Surveillance Using Fixed-Wing Miniature UAVs", Proceedings of the IEEE, vol. 94, pp. 1306 - 1324, July 2006.

[95] D.Mellinger, M.Shomin, N.Michael, V.Kumar, "Cooperative Grasping and Transport using Multiple Quadrotors", GRASP Laboratory, University of Pennsylvania.

[96] A.Franchi, C.Secchi, M.Ryll, H.H.Bulthoff, Giordano, P.R., "Shared Control : Balancing Autonomy and Human Assistance with a Group of Quadrotor UAVs," IEEE Robotics & Automation Magazine, vol.19, pp.57-68, September 2012.

[97] M. Iskandarani, S. N. Givigi, G. Fusina, A. Beaulieu, "Unmanned Aerial Vehicle formation flying using Linear Model Predictive Control", 2014 8th Annual IEEE Systems Conference (SysCon), pp. 18 – 23, Ottawa, ON, April 2014.

[98] D. Zhou and M. Schwager, "Virtual Rigid Bodies for Coordinated Agile Maneuvering of Teams of Micro Aerial Vehicles", 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 1737 – 1742, Seattle, WA, May 2015.

[99] F. Liao , X. Dong, F. Lin, "Robust Formation and Reconfiguration Control of Multiple VTOL UAVs: Design and Flight Test", 2014 22nd Mediterranean Conference of Control and Automation (MED), pp. 1440 – 1445, Palermo, June 2014.

[100] A. Chamseddine, Y. Zhang, C. A. Rabbath, D. Theilliol, "Trajectory planning and replanning strategies applied to a quadrotor unmanned aerial vehicle", Journal of Guidance, Control, and Dynamics, Vol. 35, No. 5, pp. 1667-1671, 2012.

[101] X. Dong, B. Yu, Z. Shi, Y. Zhong, "Time-Varying Formation Control for Unmanned Aerial Vehicles: Theories and Applications", IEEE Transactions on Control Systems Technology, Vol. 23, pp. 340 – 348, January 2015.

[102] U. Pilz, H. Werner, "Convergence Speed in Formation Control of Multi-Agent Systems -A Robust Control Approach", 52nd IEEE Conference on Decision and Control, pp. 6067 – 6072, Firenze, December 2013.

[103] J. D. Schaffer, "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms ", Conference: Proceedings of the 1st International Conference on Genetic Algorithms, July 1985.

[104] C. M. Fonseca and P. J. Fleming, " Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization", Proceedings of the Fifth International Conference in Genetic Algorithms, July 1993.

[105] N. Srinivas and K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithms," Evolutionary Computing, vol. 2, no. 3, pp. 221–248, Fall 1995.

[106] Horn J, Nafpliotis N, Goldberg D.E, "A niched Pareto genetic algorithm for multiobjective optimization", Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, vol. 1, pp. 67–72, 1994.

[107] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2):182 – 197, April 2002

[108] Zitzler, E. and L. Thiele. "An evolutionary algorithm for multiobjective optimization: The strength pareto approach. Technical Report 43", Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, May 1998.

[109] R. L. Galvez , E. P. Dadios , A. A. Bandala ,"Path Planning for Quadrotor UAV Using Genetic Algorithm", IEEE International Conference Humanoid, Nanotechnology, Information Technology Communication and Control, Environment and Management (HNICEM), pp. 1-6, Philippines, November 2014.

[110] H. Li, L. Wang, S. Pang, M. Towhidnejad, "Path-finding Algorithm for Ground Multiple Sensor Nodes Detection of Quad-rotor-typed UAV", 10th International Conference on Information Technology, pp. 477 – 482, Las Vegas, April 2013.

[111] I.K.Nikolos, K.P.Valavanis, N.C.Tsourveloudis, A.N. Kostaras, "Evolutionary Algorithm Based Offline/Online Path Planning for UAV Navigation," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 33, pp. 898-912, December 2003.

[112] E.Besada Portas, L. de la Torre, J. M. de la Cruz and B. de Andrés-Toro, "Evolutionary trajectory planner for multiple UAVs in realistic scenarios," IEEE Transactions on Robotics, vol. 26, pp. 619-634, August 2010.

[113] V. Roberge, M. Tarbouchi, G. Labonte, " Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning," IEEE Transactions on Industrial Informatics, vol. 9 , pp. 132 - 141, February 2013.

[114] S. Mittal, K. Deb, "Three-dimensional offline path planning for UAVs using multiobjective evolutionary algorithms", IEEE Congress on Evolutionary Computation, pp. 3195 – 3202, Singapore, September 2007.

[115] S. Kukkonen, J. Lampinen, "Ranking-Dominance and Many-Objective Optimization", IEEE Congress on Evolutionary Computation, Singapore, pp. 3983 - 3990, September 2007.

[116] Z.He, G.G. Yen, J. Zhang, " Fuzzy-Based Pareto Optimality for Many-Objective Evolutionary Algorithms," IEEE Transactions on Evolutionary Computation, vol. 18, pp. 269 - 285, April 2014.

[117] M. Koppen, R. Vicente-Garcia, and B. Nickolay, "Fuzzy-Pareto dominance and its application in evolutionary multi-objective optimization," International Conference on Evolutionary Multi-Criterion Optimization, Mexico,pp. 399–412, March 2005.

[118] M. Nasir, A. K. Mondal, S. Sengupta, S. Das, and A. Abraham, "An Improved Multiobjective Evolutionary Algorithm based on Decomposition with Fuzzy Dominance", IEEE Congress on Evolutionary Computation, New Orleans, LA, USA, pp. 765–772, June 2011.

[119] L. Batista, F. Campelo, F. Guimaraes, and J. Ramirez, “A comparison of dominance criteria in many-objective optimization problems,” in IEEE Congress on Evolutionary Computation, pp. 2359–2366, Barcelona, Spain, 2011.

[120] M. Garza-Fabre, G.T. Pulido, and C. A. Coello, “Alternative fitness assignment methods for many-objective optimization problems,” 9th International Conference, Evolution Artificielle, Strasbourg, France, pp. 146–157, October 2009.

[121] J. Horn, N. Nafpliotis, and D. E. Goldberg, “A niched Pareto genetic algorithm for multiobjective optimization,” in IEEE Congress on Evolutionary Computation, Orlando, Florida, pp. 82–87, Jun 1994.

[122] K. Deb, A. Pratap, S. Agarwal, and T.Meyarivan, “A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II,” IEEE Trans. Evol. Comput.,vol. 6, pp. 182–197, Apr. 2002.

[123] E.Zitzler and L.Thiele, “Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach,” IEEE Transactions on Evolutionary Computation, vol. 3, pp. 257–271, Aug. 1999.

[124] E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization,” in Proc. Evolutionary Methods Des. Optimisation Control, 2002, pp. 95–100.

[125] S. F. Adra, P.J. Fleming, "A Diversity Management Operator for Evolutionary Many-Objective Optimisation," Evolutionary Multi-Criterion Optimization, 5th International Conference, Nantes, France, April 7-10, 2009.

[126] M.Li, S.Yang, X.Liu, "Shift-Based Density Estimation for Pareto-Based Algorithms in Many-Objective Optimization,"IEEE Transactions on Evolutionary Computation, vol. 18, pp. 348 - 365, June 2014.

[127] S. F. Adra and P. J. Fleming, “Diversity management in evolutionary many-objective optimization,” IEEE Transactions on Evolutionary Computation, vol. 15, pp. 183–195, April 2011.

[128] H. Aguirre and K. Tanaka, “Space partitioning with adaptive-ranking and substitute distance assignments: A comparative study on many-objective MNK-landscapes,” in Proc.11th Annual Conference Genetic Evolutionary Computation, pp. 547–554,2009.

[129] X.Zou, Y.Chen, M.Liu, L.Kang, " A New Evolutionary Algorithm for Solving Many-Objective Optimization Problems," IEEE Transactions on Systems, Man, and Cybernetics, Part B, Cybernetics, vol. 38, pp. 1402-12, Oct. 2008.

[130] S.Yang, M.Li, X.Liu, J.Zheng, "A Grid-Based Evolutionary Algorithm for Many-Objective Optimization," IEEE Transactions on Evolutionary Computation, vol. 17, pp. 721 - 736, October 2013.

[131] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, “Combining Convergence and Diversity in Evolutionary Multiobjective Optimization,”Evolutionary Computation, vol.10, pp. 263–282, September 2002.

[132] Y. Zhou and J. He, “Convergence Analysis of a Self-Adaptive Multiobjective Evolutionary Algorithm Based on Grids,”Information Processing Letters,vol. 104, pp. 117–122, November 2007.

[133] H.K. Singh, A. Isaacs, T. Ray," A Pareto Corner Search Evolutionary Algorithm and Dimensionality Reduction in Many-Objective Optimization Problems," IEEE Transactions on Evolutionary Computation, vol. 15, pp. 539 - 556, August 2011.

[134] H.Wang, X.Yao,"Corner Sort for Pareto-Based Many-Objective Optimization," IEEE Transactions on Cybernetics, vol. 44, pp. 92 - 102, January 2014.

[135] R.Wang, R.C. Purshouse, P.J. Fleming," Preference-Inspired Coevolutionary Algorithms for Many-Objective Optimization," IEEE Transactions on Evolutionary Computation, vol. 19, pp. 474 - 494, August 2013.

[136] Sato, H., Aguirre, H. E., Tanaka, K, "Pareto partial dominance MOEA and hybrid archiving strategy included CDAS in many-objective optimization", IEEE Congress on Evolutionary Computation, Barcelona, pp. 1 - 8, February 2010.

[137] X.Guo, Y.Wang, X.Wang, "Using Objective Clustering for Solving Many-Objective Optimization Problems," Mathematical Problems in Engineering, vol. 2013, May 2013.

[138] K.Deb, D.K.Saxena, "On Finding Pareto-Optimal Solutions through Dimensionality Reduction for Certain Large-Dimensional Multi-Objective Optimization Problems," Indian Institute of Technology Kanpur, 2005.

[139] Y.Li, J.Zhou, H.Qin, Y.Lu, J.Yang, "Adaptive Niche Multi-objective Particle Swarm Optimization Algorithm", Fourth International Conference on Natural Computation, pp. 418 - 422, October 2008.

[140] Bouabdallah, S., Noth, A.; Siegwart, R.; “Full control of a quadrotor” ; International Conference on Intelligent Robots and Systems, pp. 153 – 158, 2007.

[141]S.W.Ekanayake, P.N.Pathirana, “Formations of robotic swarm: An artificial force based approach”, International journal of advanced robotic systems, vol.7, pp. 7-24, 2010.

[142] R. Fierro, P. Song, A. Das, V. Kumar, “Cooperative Control of Robot Formations”, Cooperative Control and Optimization, vol. 66, Series of Applied Optimization, pp. 73-93, 2002.

[143] D.Mellinger, V. Kumar, “Minimum Snap Trajectory Generation and Control for Quadrotors”, IEEE International Conference on Robotics and Automation, Shanghai, May 2011.

[144] Gaui 330X-S Instruction Manual, Internet: http://www.gauimrt.com/support/downloads/, September 2010.

[145] T580 Basic Quadcopter, Internet: http://www.uavshop.co.uk/downloads/X4-Manual-UAVshop.pdf, March 2011.