**T.C.**

**MARMARA UNIVERSITY**

**INSTITUTE FOR GRADUATE STUDIES IN**

**PURE AND APPLIED SCIENCES**

# 3-D OFFLINE PATH PLANNING FOR UNMANNED AERIAL VEHICLES BY USING METAHEURISTICS

**Işıl HASIRCIOĞLU**

**THESIS**

**FOR THE DEGREE OF MASTER OF SCIENCE**

**IN**

**COMPUTER ENGINEERING**

**SUPERVISOR**

**Assoc. Prof. Haluk TOPÇUOĞLU**

**İSTANBUL 2008**

**T.C.**

**MARMARA UNIVERSITY**

**INSTITUTE FOR GRADUATE STUDIES IN**

**PURE AND APPLIED SCIENCES**

# 3-D OFFLINE PATH PLANNING FOR UNMANNED AERIAL VEHICLES BY USING METAHEURISTICS

**Işıl HASIRCIOĞLU**

**(141100320050015)**

**THESIS**

**FOR THE DEGREE OF MASTER OF SCIENCE**

**IN**

**COMPUTER ENGINEERING**

**SUPERVISOR**

**Assoc. Prof. Haluk TOPÇUOĞLU**

**İSTANBUL 2008**

# ACKNOWLEDGEMENTS

I would like to thank to my advisor, Assoc. Prof. Dr. Haluk Topçuoğlu for his endless support and encouragement. I would also thank to Prof. Dr. Fuat İnce and Prof. Dr. Oğuz Tosun for serving my thesis committee.

I would like to thank to my dear friend, Perihan Öbek for her moral support and Fuat Geleri for his information share. I also would like to thank to our assistants Fatma Çorut Engin, Betül Demiröz and Berna Kiraz for their support whenever I need help about technical problems.

I am grateful to my unique family; my mother Vahide Hasırcıoğlu, my father Ayhan Hasırcıoğlu and my fiancé Dindar Öz for their endless love, patient and great support during my thesis like as during my whole life.

**June, 2008** **Işıl HASIRCIOĞLU**

# TABLE OF CONTENTS

**ACKNOWLEDGEMENTS** ..... **1**

**TABLE OF CONTENTS** ..... **2**

**ÖZET** ..... **4**

**ABSTRACT** ..... **5**

**LIST OF SYMBOLS** ..... **6**

**ABBREVIATIONS** ..... **7**

**LIST OF FIGURES** ..... **8**

**PART I INTRODUCTION** ..... **12**

**PART II PROBLEM DEFINITION** ..... **15**

**II.1. UAV CHARACTERISTICS** ..... **16**
**II.2. PATH PLANNING PROBLEM for UAVs** ..... **17**
**II.3. RELATED WORK** ..... **17**

**PART III METAHEURISTIC TECHNIQUES** ..... **21**

**III.1. GENETIC ALGORITHMS** ..... **21**
III.1.1. Genetic Algorithm Components ..... 22

**III.2. PARTICLE SWARM OPTIMIZATION** ..... **24**
III.2.1. The Algorithm ..... 24

III.2.2. PSO parameter control ..... 25

**III.3. HYPER-HEURISTICS** ..... **26**
III.3.1. Level 0 (Problem Representation) ..... 29

III.3.2. Level 1 (Low-Level heuristics) ..... 29

III.3.3. Level 2 High-level heuristic (Hyper-heuristic) ..... 29

**PART IV PROPOSED FRAMEWORK** ..... **35**

**IV.1. TERRAIN REPRESENTATION** ..... **35**
**IV.2. PATH REPRESENTATION** ..... **37**
**IV.3. OBJECTIVES of the PATH PLANNING PROBLEM** ..... **39**
**IV.4. GA-BASED FORMULATION of the PROBLEM** ..... **47**
IV.4.1. Representation of individuals ..... 47

IV.4.2. Initial Population ..... 48

IV.4.3. Parent Selection Mechanism ..... 48

IV.4.4. Survivor Selection Mechanism ..... 49

IV.4.5. Crossover and Mutation Operators ..... 49
**IV.5. PARTICLE SWARM OPTIMIZATION ..... 55**
**IV.6. HYPER-HEURISTICS ..... 57**
IV.6.1. Problem Representation ..... 57
IV.6.2. Low-level Heuristics ..... 57
IV.6.3. Hyper-Heuristic Techniques ..... 59
**PART V EXPERIMENTAL STUDY ..... 60**
**V.1. TERRAIN GENERATION ..... 60**
V.1.1. Hill Algorithm ..... 60
V.1.2. Midpoint Displacement Algorithm ..... 62
**V.2. PRE-EXPERIMENTATION PHASE ..... 64**
V.2.1. Experiments for Setting GA Parameters ..... 64
V.2.2. Experiments for Setting PSO Parameters ..... 72
V.2.3. Experiments for Setting HH Parameters ..... 76
**V.3. COMPARISON STUDY ..... 79**
**PART VI CONCLUSION AND FUTURE WORK ..... 85**
**REFERENCES ..... 86**
**RESUME ..... 91**

# ÖZET


## İNSANSIZ HAVA TAŞITLARINA YÖNELİK OLARAK EVRİMSEL ALGORİTMALAR YARDIMI İLE OFFLINE GÜZERGAH PLANLAMA

İnsansız hava araçları, uzaktan kontrol edilen veya önceden programlanmış uçuş planlarına göre otomatik olarak hareket eden uçaklardır. İnsansız hava araçları; hava keşifleri, deniz ve dağlardaki arama kurtarma çalışmaları, hava fotoğrafçılığı ve haritacılığı, yangın tespiti ve trafik kontrolü gibi çeşitli amaçlar için kullanılmaktadır. İnsansız hava araçlarının otomatik çalışmalarını sağlayabilmek için uzun süre insan desteği olmadan çalışma özelliği sağlayacak kontrol sistemleri geliştirilmesi gerekmektedir. Engellere çarpmadan en uygun yolu izleyecek yollar üreten yol planlayıcıları, insansız hava araçlarına otomatik çalışma özelliği kazandırabilmektedirler. Uçuş planının optimize edilmesi, değişken ortam koşullarının ve hava şartlarının göz önüne alınması gerektiği çok amaçlı bir problemdir.

Bu projede üç boyutlu ortamlarda otomatik olarak hareket eden insansız hava araçları için bir offline (sabit ortam ve kısıtlarda) yol planlayıcısı geliştirdik. Geliştirilen yol planlayıcısı; arazi yüzeyine çarpmayan, belli bir görüş mesafesinde izin verilen minimum dönme açısından büyük açılarla, kısıtlanan minimum ve maksimum yükseklikler arasında belirtilen tehlikeli bölgeden uzak bir seyir izleyen en kısa yolu bulmayı hedeflemektedir.

Bu yol planlayıcısını geliştirmek için farklı problemler için kullanılan metasezgisel (metaheuristic) yöntemler (Genetik Algoritmalar, Parçacık Sürü Eniyileme ve HyperHeuristics) kullandık. Farklı arazi tipleri için gerçekleştirdiğimiz deneyler neticesinde, Genetik Algoritmaların çoğu arazi için diğer yöntemlerden daha iyi sonuçlar verdiğini gördük. Bununla birlikte HyperHeuristics yönteminin zor arazi tiplerinin çoğunda diğer yöntemlerden daha iyi sonuçlar verdiğini gözlemledik. Parçacık Sürü Eniyileme yönteminin ise diğer yöntemlerle karşılaştırıldığında iyi sonuçlar vermezken mimimum çalışma zamanına sahip olduğu gözlendi.


**Haziran, 2008** **Işıl HASIRCIOĞLU**

# ABSTRACT

## 3-D OFFLINE PATH PLANNING FOR UNMANNED AERIAL VEHICLES BY USING METAHEURISTICS

Unmanned Aerial Vehicle (UAV) is an aircraft which is remotely controlled or can fly autonomously according to the pre-programmed flight plans. UAVs are used for many missions including weather reconnaissance, search and rescue assisting operations in sea and mountains, aerial photographing and mapping, fire detection, traffic control. Autonomous operation of UAVs requires the development of control systems that can work without human support for long time periods. The path planners which generate collison-free and optimized paths are needed to give autonomous operation capability to the UAVs. The optimization of the flight path is a multi-objective problem dealing with variable terrain features as well as dynamic environment conditions**.**

In this study, we develop an offline path planning framework for unmanned aerial vehicle navigation in 3-D environments. Our framework targets to find the shortest path and / or a flight envelope in a given line of sight by avoiding terrain collisions, traveling on a path which provides within the restricted minimum and maximum distances to the terrain, traveling far from the specified threat zones, and maneuvering with an angle greater than allowable minimum curvature radius.

We present various metaheuristic techniques (Genetic Algorithms, Particle Swarm Optimization and Hyper-Heuristics) in order to construct the paths for UAV navigation. Based on the comparative study on terrains with various characteristics, it was observed that our GA-based method outperforms the other two heuristics with respect to quality of solutions (given in total cost of a constructed path including the penalties of all constraints) for most of the terrains examined. On the other hand, Hyper-Heuristics outperforms other methods for more than half of the hard terrains. Although, the PSO method does not perform well when compared with the other techniques, it requires the minimum running time among them to generate a feasible path.

**June, 2008** **Işıl HASIRCIOĞLU**

# LIST OF SYMBOLS

$V_i[k]$ : Velocity vector for particle $i$ at time $k$

$P_i[k]$ : Particle position

$Pbests_i[k]$ : Best position of the particle in its past experience

$Gbest[k]$ : Best position among all particles in the population

$N_j$ : Individual low level heuristic

$f_1(N_j)$ : Individual performance of the heuristic $N_j$

$I_n(N_j)$ : Change in the fitness function the $n^{th}$ last time heuristic $N_j$ was called

$T_n(N_j)$ : The amount of time taken the $n^{th}$ last time heuristic $N_j$ was called

$f_2(N_j, N_k)$: Aggregate performance of the heuristic $N_j$ and consecutive heuristic $N_k$

$I_n(N_j, N_k)$: Change in the fitness function the $n^{th}$ last time heuristic $N_k$ was called after heuristic $N_j$

$f_3(N_j)$ : Amount of time elapsed since the heuristic $N_j$ was last called

$f(N_j)$ : Choice function

$C(u)$ : B-Spline curve representation

$N_{i,k}$ : B-Spline curve basis function

$f_i$ : Cost function

$f_1$ : Collision penalty

$f_2$ : Safety distance penalty

$f_3$ : Curvature radius penalty

$f_4$ : Threat zone penalty

$f_5$ : Shortest path penalty

$f_6$ : Line of sight penalty

# ABBREVIATIONS

**UAV** **:** Unmanned Aerial Vehicle

**GA** **:** Genetic Algorithms

**PSO** **:** Particle Swarm Optimization

**HH** **:** Hyper-Heuristics

# LIST OF FIGURES


**PAGE NUMBER**

**Figure III.1.1**The general scheme of a genetic algorithm [3] 22

**Figure III.1.2** The general scheme of a genetic algorithm as flow-chart [3] 23

**Figure III.2.1** Particle Swarm Optimization Algorithm 25

**Figure III.3.1** General hyper-heuristic framework [41] 27

**Figure III.3.2** Hyper-heuristic algorithm [39] 28

**Figure III.3.3** Hyper-heuristic Framework [39] 28

**Figure III.3.4** Simple Random pseudocode 30

**Figure III.3.5** Random Descent pseudocode 30

**Figure III.3.6** Random Permutation pseudocode 30

**Figure III.3.7** Random Permutation Descent pseudocode 31

**Figure III.3.8** Greedy Approach pseudocode 31

**Figure III.3.9** Straight Choice 33

**Figure III.3.10** Ranked Choice 34

**Figure III.3.11** Roulette Choice 34

**Figure IV.1.1** Heightmap represented by regularly-spaced grids [47] 35

**Figure IV.1.2** The cells of the height map & view of the terrain generated by that map [48] 36

**Figure IV.2.1** B-spline curve representation [51] 38

**Figure IV.3.1** Collision penalty calculation pseudocode 40

**Figure IV.3.2** Safety distance penalty pseudocode 42

**Figure IV.3.3** Max distance penalty pseudocode 42

**Figure IV.3.4** Schematic representation of the curvature angle 43

**Figure IV.3.5** Curvature radius penalty pseudocode 44

**Figure IV.3.6** Threat zone penalty calculation pseudocode 44

**Figure IV.3.7** Shortest path calculation pseudocode 45

**Figure IV.3.8** LOS penalty calculation pseudocode 46

**Figure IV.4.1** Individual Representation by B-Spline Curve 47

**Figure IV.4.2** The Paths with Different Number of Control Points 48

**Figure IV.4.3** Knowledge-based crossover pseudocode 50

**Figure IV.4.4** Crossover Operators 51
**Figure IV.4.5** Knowledge-based update mutation pseudocode 53
**Figure IV.4.6** Knowledge-based insert mutation pseudocode 53
**Figure IV.4.7** The types of Mutation Operators 54
**Figure IV.5.1**General scheme for PSO Algorithm as pseudocode 55
**Figure IV.5.2** Velocity calculation in PSO Algorithm for path planning problem 56
**Figure IV.5.3** Velocity vs control point location 56
**Figure IV.6.1** Hyper-heuristic pseudocode 58
**Figure V.1.1** Multiple Hills [55] 61
**Figure V.1.2** A terrain generated by Hill Algorithm 61
**Figure V.1.3** Iterations of displacements in one dimension [57] 62
**Figure V.1.4** A terrain generated by Midpoint displacement algorithm 62
**Figure V.1.5** Sample terrains used in our experiments, above Terrain1 and Terrain2 generated by midpoint displacement, below Terrain3 and Terrain4 generated by hill algorithm 63
**Figure V.2.1** Lowest cost value of each generation for four terrains 66
**Figure V.2.2** Average cost value of each generation for four terrains 68
**Figure V.2.3** Feasibility of the best solution in each generation by considering four constraints of the unified objective 69
**Figure V.2.4** Cost value of the best solution by varying the number of control points for terrain3 70
**Figure V.2.5** Cost value of the best solution by varying the number of control points for terrain2 70
**Figure V.2.6** Performance of four different mutation operators on different terrains 71
**Figure V.2.7** Performance of three different crossover operators on different terrains 72
**Figure V.2.8** Lowest cost value of each generation for four terrains 75
**Figure V.2.9** Average cost value of each generation for four terrains 76
**Figure V.3.1** New Terrains for Comparison Study (numbered as Terrain5, Terrain6, Terrain7, Terrain8, Terrain9, Terrain10 respectively) 80
**Figure V.3.2** The Means and Distributions of the Normalized Cost Values for the Algorithms for an Easy Terrain (Terrain6) 82

**Figure V.3.3** The Means and Distributions of the Normalized Cost Values for the Algorithms for an Hard Terrains (Terrain10) 83
**Figure V.3.4** Sample Paths for Sample Terrains 84

# LIST OF TABLES


**PAGE NUMBER**

**Table I.1-1** UAV Characteristics[9] 16

**Table IV.3-1** Constraints in the Path Planning Problem 39

**Table V.2-1** GA parameter setting 64

**Table V.2-2** Final settings for path planner Genetic Algorithm 72

**Table V.2-3** PSO Parameter Setting 73

**Table V.2-4** Final settings for path planner PSO algorithm 73

**Table V.2-5** HH Parameter Setting 77

**Table V.2-6** Average cost values obtained by HH methods 78

**Table V.2-7** Final settings for path planner HH algorithm 79

**Table V.3-1** Difficulty Levels of Terrain Structures Used in the Experiments 81

**Table V.3-2** The Comparison of Average Cost Values for the Proposed Algorithms 81

**Table V.3-3** The Standard Deviation of Cost Values of the Algorithms 82

**Table V.3-4** The Average Running Times of the Algorithms in Miliseconds 84

# PART I

## INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are remotely-controlled or self controlled aircrafts which can carry cameras or sensors to gather information about the environment. They have been used both in commercial and military applications for various missions including weather reconnaissance, target acquisition, border patrol, search and rescue assisting operations and environmental monitoring. Teaming of UAVs also provide cooperative tasking of UAVs to perform more complex tasks [1].

An autonomous operation of UAVs needs a control system which decides the actions and maneuvers of the vehicle to conform its mission. The use of these automated systems reduces the human personnel requirement which should operate the vehicle motion. It can operate longer time and in higher altitudes without human support. Moreover, it reduces the financial expenses for the human personnel by assigning them for the other tasks. Once the vehicle is programmed by the control system, it can operate its mission automatically without any interference.

Path planning of a UAV is a multi-objective optimization problem which takes into account various constraints including technical, environmental and mission specific constraints. Since UAV has many special characteristics and technological constraints. Moreover, UAVs have different missions in different environmental conditions. Therefore, the path planner also has to deal with various conditions to travel between the specified source and destination points in a correct and efficient way.

The aim of this thesis is to develop an offline path planning framework for unmanned aerial vehicle navigation in 3-D terrain environments based on several metaheuristics. We represent paths with B-Spline curves, which are continuous parameterized curves [2]. The major constraints of the paths that are generated in our framework include: feasible paths without terrain collision, paths within the desired minimum and maximum distance to the terrain, paths that provide UAV to maneuver with an angle greater than allowable minimum curvature radius, paths that avoid

threat zones in the terrain. By considering these constrains, our path planner targets to minimize the length of the path from source to destination or it targets to provide a flight envelope in a given line of sight.

The offline path planning problem for UAVs in 3-D environment has high computational complexity to find the optimum path by dealing with several constraints. Due to the problem is NP-complete which is hard to solve the problem in reasonable time; heuristic techniques that target to develop near-to-optimum or good results in reasonable times have been used in our study. Therefore, as part of this study, we target to analyze the performance of various metaheuristic techniques for UAV path planning including Genetic Algorithms, Particle Swarm Optimization and Hyper-Heuristics.

Genetic Algorithms (GAs) are nature-inspired population-based metaheuristic optimization algorithms [3]. It uses techniques inspired by natural evolution such as selection, crossover and mutation. In our study, we have also designed and implemented problem-specific operators (one crossover operator and three mutation operators) in order to improve the quality of solutions for the generated terrains.

Particle Swarm Optimization (PSO) is another population-based metaheuristic which is inspired by bird flocking and fish schooling (teaching fish to survive and reproduce) [4]. We have designed a PSO system which deals with the path planning problem of UAVs. The candidate solutions are represented by the particles in the swarm. Since PSO method is easy to implement and adapt the problem, we have considered this heuristic to solve the UAV path planning problem.

Hyper-Heuristics (HH) is another metaheuristic technique which aims to find the best heuristic among various heuristics [5]. It has many low-level heuristics which are designed to solve the problem. By HH, the best suitable heuristic is selected by the selection mechanisms and it's applied to the population. HH has been recently used for multi-objective problems, since it is easy to implement by defining different heuristic methods which are also easy to implement and produces efficient results in reasonable times. We have defined different mutation types as low-level heuristics and by the selection mechanisms proposed in the literature [6], the mutation type to be applied to the current solution is chosen.

In our study, the performance of the three metaheuristics is measured with various terrains that have different characteristics. As part of this study, we use a tool for generating 3-D terrains in which the generated terrains differ with respect to

terrain generation algorithm considered and the corresponding parameters. The experimental study is performed in two phases. First phase is to determine the best or most suitable values for parameters considered in each metaheuristic. Specifically, it is important to identify the value of the crossover rate, the crossover type, the mutation rate, and the mutation type for GA, the swarm size for PSO, the selection mechanism and the acceptance criteria for the hyperheuristics. Moreover, the number of control points for terrains that have different difficulty level have been tested to get the lowest cost value, namely the best performance. The second phase of the comparison study measures the quality and the cost of the solutions generated by three metaheuristics (GA, PSO and HH) for terrains with various characteristics. In our experiments, it's observed that the GA generally performs better results (in terms of total costs) than other methods for most of terrains; additionally, HH gives best results for some of the harder terrain structures.

This thesis organized as follows. The Part II presents the design issues of UAVs, which is followed by related work on UAV path planning problem in the literature. In Part III, we present brief information on the three metaheuristic techniques considered in this study. Our proposed framework for offline path planning is introduced in Part IV. After specifying terrain and path representations of the problem and defining the objectives that should be dealt with, the implementation details of the proposed algorithms are explained by providing pseudocodes and flowcharts. Part V presents the experimental study including the experiments which are to fine-tuning the parameters and the comparison study which works on each algorithm in detail. Finally, conclusion and future work is presented in Part IV.

# PART II

## PROBLEM DEFINITION

Path planning is a widespread research topic in many areas including robot navigation to avoid obstacles and flight navigation to avoid bad weather conditions, radar detection or other threats. As the usage of the autonomous vehicles has been increasing in different areas, the path planning strategies to route the vehicles have gained importance.

The path planning generates a trajectory from an initial position to the specified destination point without colliding any obstacles by performing specific objectives and completing the vehicle's tasks. In general a path planning problem consists of initial location of the vehicle, a set of targets to be observed or attacked, a set of threats, a set of base or goal locations, a model of the environment given and a trajectory for the vehicle that is to find through the set of observation or attack region, which optimizes a performance function subject to the set of constraints [7].

The path planning problem may be categorized as offline path planning in which the vehicle travels in a completely known static environment and online path planning in that some conditions about the environment is not known. In the first case; the starting and ending points, the obstacles or threats to avoid, the tasks to be accomplished are all known at the beginning phase of the path planning. The complete plan of the route may be constructed by getting these static conditions. In the latter case, some conditions about the environment are not known initially. The obstacles may appear along the travel of the vehicle like in real world situations. For instance, weather conditions may change while the vehicle travels around or the final target may be altered due to the mission revision. Both offline and online path planning problems may be considered for robotics as well as aerial vehicles. In this thesis, offline path planning for Unmanned Aerial Vehicles (UAVs) has been studied; and online version is out of scope of this thesis [8].

## II.1. UAV CHARACTERISTICS

The UAV is an acronym for Unmanned Aerial Vehicle, which is an aircraft with no pilot on board. UAVs can be controlled remotely by a pilot at a ground control station or can fly autonomously based on pre-programmed flight plans or more complex dynamic automation systems. UAVs may be categorized into different types [1]:

- Target and decoy used to simulate an enemy aircraft or missile,
- Reconnaissance providing battlefield intelligence,
- Combat used to get attack capability for high-risk missions,
- Logistics designed for cargo and logistics operations,
- Research and development used for further development and research of UAV technology,
- Civil and Commercial UAVs designed for civil and commercial applications.

UAV characteristics specified in UAV Annual Report by Office of the Under Secretary of Defense, Defense Airborne Reconnaissance Office (DARO) [9] are shown in Table II.1-1.

| **UAV Characteristics** |
|---|
| Altitude: maximum, operating (km),<br>Endurance: maximum (hrs),<br>Radius of Action,<br>Speed,<br>Climb Rate,<br>Deployment Needs |
| Propulsion: Engine,<br>Weight,<br>Dimensions,<br>Avionics,<br>Launch&Recovery,<br>Guidance Control |
| Sensor,<br>Data Link,<br>C2 Link |
| System Composition,<br>Contractors |

**Table II.1-1 UAV Characteristics[9]**

UAVs have been recently used in commercial applications as well as in military areas [10] for different missions including weather reconnaissance, search and rescue assisting operations in sea and mountains, aerial photography and mapping, fire detection and traffic control. The most important reasons of the popularity of the UAVs are low force need, no vital risk by performing the most hazardous missions and long endurance.

### II.2. PATH PLANNING PROBLEM for UAVs

Autonomous operation of UAVs requires the development of control systems that can work without human support for long time periods. The vehicles are required to make both low-level control decisions, such as path planning, and high-level decisions, such as cooperative task assignment. Task assignment is crucial for designing successful missions in heavily defended environments while the path planners which generate collision-free and optimized paths are needed to give autonomous operation capability to the UAVs. Both aspects of the fleet problem of the UAVs build up the optimal flight by deciding resource allocation and determining the best trajectory.

The path planning for UAVs is a multi-objective optimization problem which takes into account three main constraints: *technical constraints*, *environmental constraints* and *mission specific constraints*.

The major technical constraints include minimum turning radius, velocity of the vehicle, fuel consumption and flight time, which are based on UAV's technical characteristics. The environmental constraints include topology and other characteristics of the terrain and weather conditions (such as rain, snow etc.). The mission-specific constraints for the path may change according to the objective of the flight, such as the path with the maximum region coverage or specific region coverage, the shortest path from the given source to the given destination, a path which has sweeping flight, a path which has a flight envelope in a given line of sight, or finally a path which provides specified minimum and maximum distance to the terrain.

### II.3. RELATED WORK

Path planning of UAVs have been considered for different objectives and different approaches in the literature.

The A-star algorithm (which is a heuristic graph search method) has been applied to find shortest path for the UAV [11].

Potential fields-based algorithm has been implemented in a task planning and control system developed for multi UAVs, to be able to detect the terrain's burnability risk [12]. Each cell of the ground is associated to a potential, initiated to a maximum value, and decreasing with time according to a specific law.

There have been probabilistic methods to explore the diversities in the non-stationary environments. The principle of these methods is to construct a probabilistic map which defines the risk of threat for the vehicles. For instance, the depth map constructed in [13] provides information about the distance between the agent and the obstacles. The conditional probability that a UAV is defeated by a threat under the condition that it follows a path is defined and formulated based on the probabilistic map of the area [14]. The probability of zero detections are maximized in case of circular threat zones as well as nested convex polygonal threat zones which is a very general way to model the threat zones [15]. Paths minimizing the flying time are also considered while providing minimum threat detection risk.

There are also MILP (Mixed Integer Linear Programming) based approaches in the literature for UAV navigation [16, 17] where MILP is a mathematical programming framework. The vehicle is characterized by a discrete, linear model in a coordinate frame and additional linear inequalities representing dynamic and kinematic safety constraints [16]. In a multi-vehicle team control system [17], trajectory optimization is handled by using a MILP-based receding horizon planner, which provides the arrival of the UAVs at the target in time.

Dynamic programming techniques are also considered for UAV navigation, such as task assignment [18] and path planning [19]. Although the exact dynamic Programming algorithm generates optimal solutions in a risky environment, two different DP approximation methods (the one-step and two-step lookahead) reducing computation time for larger problems have been proposed [18]. A path planning in an unknown environment problem has been modeled as an adaptive control of a Markov Decision Process (MDP) [19], where MDP is a process characterized by a set of states. In each state there are several actions from which the decision maker must choose [20]. The transition probabilities of the underlying Markov chain are estimated and the control is applied based on the most recent estimate of the transition probabilities, known as certainty equivalence principle.

Vision-based approaches which get environment information from the camera and generate collision avoidance algorithm for UAV navigation have been developed by different studies. One of them has used an implementation of an Unscented Kalman Filter (UKF) which generates estimates of the vehicle state and obstacle position for the trajectory generator which computes a collision free trajectory to be followed by the small UAVs navigated in the unsurveyed environments [21]. Another study [22] applies a computer vision algorithm for collision detection based on feature density distribution (FDD) analysis which requires minimum camera calibration and works well with low image quality.

Many evolutionary algorithm-based methods such as Differential Evolution, Particle Swarm Optimization, Genetic Algorithms have been applied to the UAV navigation problem. Differential Evolution (DE) is a method of mathematical optimization of multidimensional functions and belongs to the class of evolution strategy optimizers [23]. DE is an efficient method to solve the coordinated UAV path planning problem [24]. Considering a number of UAVs at different known initial locations, 2-D trajectories formed by successive way-points with a specified velocity distributions along each vehicle's trajectory and arriving a common target by satisfying route constraints has been generated by the DE based method.

Particle Swarm Optimization (PSO) which is a stochastic, population-based algorithm has been also applied to the UAV navigation problems. The autopilot design of a tilt-rotor UAV developed by Korea Aerospace Research Institute (KARI) has been established by using PSO method which handles nonlinear dynamics and provides easy modification in the structure of autopilot systems [25]. The path planning of UAVs using PSO method has been studied in 3-D environments [26] as well as 2-D environments [27]. Both researches ensure avoiding threat zones by representing the UAV paths as swarm particles.

Evolutionary approaches have been used to get cost function values to decide the probability of a given path successfully traversing the uncertain environment [28]. Two different methods to compute cost function have been proposed; one is on the exact probability of intersection of the vehicle with an obstacle, another one is a more computationally tractable approximation technique namely field integral approximation.

Genetic algorithm gives successful results for 3-D UAV path planning problem. The basic idea behind the planner is to generate a set of paths that are

feasible solutions to the problem. After simulating a natural selection process by applying combinations and modifications, better solutions among their generation are retained. These solutions constitute a new population which consists of new candidate paths. The process is repeated until a solution satisfies certain criteria [3].

Different encoding schemes have been used to represent the candidate path solutions. Since the straight lines are not sufficient and efficient representation, types of geometries have been selected to represent trajectories. One is segment representation which consists of many segments. The encoding of the paths is a sequence of simple splines. There are two types of segments including straight segments to represent climbs, descents and lines, and curved segments to represent turns [30]. On the other hand, B-Spline curve representation is applied to represent the flight path of the UAVs in [31] where they generate terrain using only mathematical functions which requires a large set of experimentally predefined constants.

Different path mutation mechanisms have been proposed in GA-based solutions. In [32]; different operators including mutate and propagate, crossover, go to goal, mutate and match which change the parameters curve segments have been implemented. In another study [33], non-uniform mutation is considered, in which the search space for the new gene is not fixed, but it shrinks close to the previous value of the corresponding gene as the algorithm converges.

In our study, we have considered different objectives together including constructing feasible paths without collision, within the desired safety distance, with an angle greater than minimum radius of curvature, avoiding threat zones, minimizing the length in a given line of sight for the UAV path planning problem. We have implemented three different metaheuristics including Genetic Algorithms, Particle Swarm Optimization and Hyperheuristics. In GA, three different knowledge-based mutation operators including update, delete, insertion and one knowledge-based crossover operator have been proposed to increase the quality of the solution. The proposed knowledge-based operators and basic mutation operators which do not have problem specific approaches have been used as low-level heuristics in hyperheuristic method. The comparison of these techniques has been performed by considering fitness value and running time in average.

# PART III

## METAHEURISTIC TECHNIQUES

In this chapter, we present the details of the three metaheuristic techniques considered in our path planning framework, which are Genetic Algorithms, Particle Swarm Optimization and Hyper-Heuristics.

### III.1. GENETIC ALGORITHMS

Evolutionary algorithms are nature-inspired population-based metaheuristic optimization algorithms applied in many engineering and operations research problems. Genetic Algorithms (GA) are the most widely known type of the evolutionary algorithm which is developed by John Holland in 1975 [34].

Genetic algorithm generates a random population of specific size among the possible solutions. Given an objective function, the fitness of the individuals is evaluated and a pair of candidate solutions is selected for mating. Genetic crossover and mutation operators are applied to the selected parents to generate new candidate solutions that compete with the old ones to replace in the next generation. This process continues until the specified termination condition is satisfied. Figure III.1.1 shows the general scheme of a Genetic Algorithm.

```
BEGIN
    INITIALIZE population with random candidate
solutions
    EVALUATE each candidate
    REPEAT UNTIL (TERMINATION CONDITION is satisfied)
DO
        1 SELECT parents
        2 RECOMBINE pairs of parents
        3 MUTATE the resulting offspring
        4 EVALUATE new candidates
        5 SELECT individuals for the next generation
END
```

**Figure III.1.1The general scheme of a genetic algorithm [3]**

### **III.1.1.** Genetic Algorithm Components

In this section, we give brief descriptions of a genetic algorithm's components.

**Representation of individuals:** It is a way of representing individuals by specifying a mapping from the objects forming the solutions (phenotypes) to their encodings as individuals in GA (genotypes). There are binary string, integer, floating point, permutation representations which are chosen according to the problem definition.

**Objective function:** It quantifies the optimality of the solution.

**Population:** It holds the individuals which are possible solutions.

**Initialization:** It is the first population generation phase which is generally seeded randomly or created by using heuristics to generate individuals which have better fitness values.

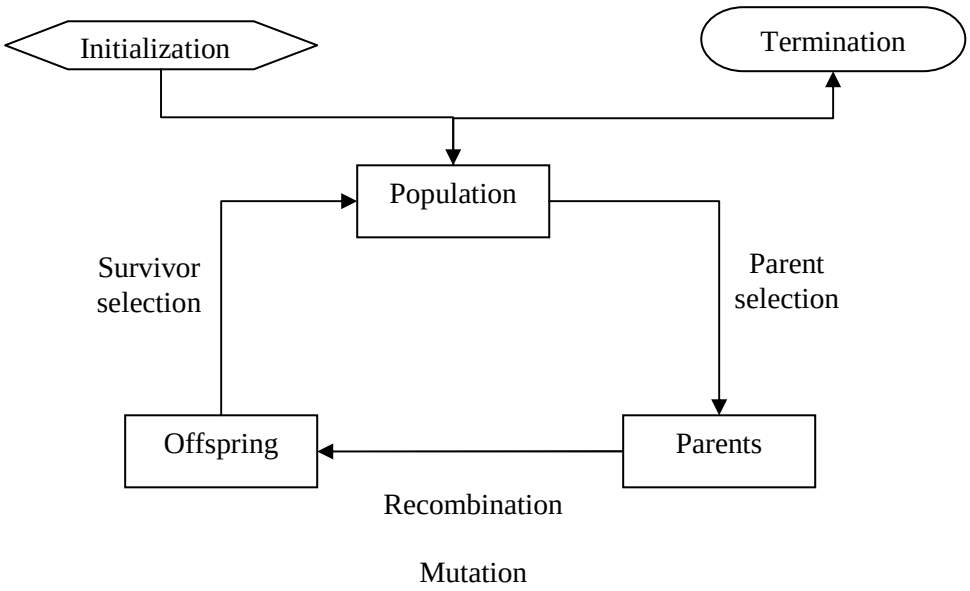



**Figure III.1.2 The general scheme of a genetic algorithm as flow-chart [3]**

**Parent selection mechanism:** It aims to select the better individuals to become parents of the next generation. There are different selection [3] including fitness proportional selection (FPS), ranking selection, roulette wheel selection and tournament selection. Mostly used ones are roulette wheel selection which is same as spinning a roulette wheel where the individual selection probabilities are reflected by the holes' sizes and tournament selection which selects individuals from the population randomly and selects the parents among the selected individuals by comparing their fitness values.

**Variation operators:** The crossover and mutation operators' aim is to create new individuals from the old ones. The crossover is resulted between two parents. The choice of what parts of each parent are combined and the way these parts are combined depends on the crossover type. The mutation is applied to a single individual and it modifies a part of the chromosome.

**Survivor selection mechanism:** It aims to select the better offspring based on their quality to survive in the population. There are different strategies including age-based replacement and fitness-based replacement strategies.

**Termination condition:** It specifies when the genetic algorithm will stop by reaching optimum level or satisfying the stopping criteria. The most common conditions used are CPU time allowed has elapsed, total number of fitness

evaluations has reached a limit, the fitness has not improved for an amount of time, the population diversity has dropped under a given threshold value.

## III.2. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995 [35], inspired by social behavior of bird flocking or fish schooling. Similar to genetic algorithms, PSO searches for optimum solution using a population of particles and its initial population is generated randomly. However, PSO does not have operators like crossover or mutation [4].

In a PSO system, the candidate solutions namely particles change their positions by flying through the multidimensional problem space. The particles follow the current optimum particles until an unchanged position has been reached or another stopping criteria is satisfied.

A PSO system combines a social-only model and a cognitive-only model [36]. According to the social-only component, individuals behave like other individuals in the population by imitating their behaviors. However cognitive-only component offers that individuals live isolated from the rest of the population and behave whatever they own like or want. An individual in PSO system behaves by considering these two different components.

### **III.2.1.** The Algorithm

As mentioned before, PSO simulates the behaviors of bird flocks; i.e., the particles searching for the optimum solution in a search space in the PSO system are like birds searching for the food in an area. There is only one optimum solution like only one food in the area. All particles do not know which solution is the optimum like all birds do not know where the food is. Therefore, the best strategy for the particles is to follow the particle which is the best fitness value in the population like following the bird nearest to the food.

PSO is initialized randomly with possible solutions called particles [4]. Each particle flies around the problem space searching for the optimum solution. A particle updates its position according to the experience gained as well as experience of neighboring particles. The particles are modified according to the following equations [37]:

$$V_i[k+1] = wV_i[k] + C_1 * Rand_1 * (Pbests_i[k]) + C_2 * Rand_2 (Gbest[k] - P_i[k]), \quad \text{(III.1)}$$

$$P_i[k+1] = P_i[k] + V_i[k] \quad \text{(III.2)}$$

where $V_i[k]$ is the velocity vector for particle $i$ at time $k$, representing the distance and the direction to be traveled by this particle from its current position; $P_i[k]$ is the particle position; $Pbests_i[k]$ is its best position in its past experience; $Gbest[k]$ is the best position among all particles in the population; $Rand_1$ and $Rand_2$ are two random functions in a range [0, 1]; $C_1$ and $C_2$ are positive constants, called the cognitive and social parameter respectively; and finally, $w$ is inertia weight to control the impact of the history of velocities on the current velocity.

```
BEGIN
    Initialize swarm with random particles
    Initialize each particle
    DO
        FOR each particle
            Calculate fitness value
            If the fitness value is better than the best
            fitness value set the current value as the best
            fitness value
        END
        Choose the particle with the best fitness value
        FOR each particle
            Calculate particle velocity
            Update particle position
        END
    WHILE termination condition is satisfied
END
```

**Figure III.2.1 Particle Swarm Optimization Algorithm**

### **III.2.2.** PSO parameter control

There are a number of parameters to be determined in a PSO implementation [38].

**The number of particles:** It is the number of possible solutions generated randomly at the beginning of the algorithm. While for most of the problems, smaller size like

10-20 is large enough to get good results; for some difficult or special problems larger particle number may be used.

**Maximum velocity:** It determines the maximum change that one particle can take during an iteration.

**Learning factors:** $C_1$ and $C_2$ are cognitive and social parameters respectively and usually $C_1$ equals to $C_2$ and ranges from [0, 4]

**The stop condition:** It shows the maximum number of iterations the PSO executes and the minimum error requirement.

**Inertia weight:** It is employed to control the impact of the previous history of velocities on the current velocity. A larger inertia weight w facilitates global exploration (searching new areas) while a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area. Suitable selection of the inertia weight w can provide a balance between global and local exploration abilities and thus require less iteration on average to find the optimum [38].

### III.3. HYPER-HEURISTICS

Hyper-heuristics (HH) is an emerging meta-heuristic technique which works by searching the right heuristic or algorithm in a given problem. The term "*hyper*heuristics" is to describe the process of using heuristics to choose heuristics to solve the problem by operating on a search space of heuristics [39, 41, 43].

For example, the Evolutionary Computation based approaches (such as Gas) for path planning problem consider populations of possible paths and the main idea is that the population will evolve along a number of generations with the aim of finding the best solution to the problem. However, a hyper-heuristic evolutionary approach deals with a population of heuristics for the path planning problem and these heuristics will evolve along a number of generations.

Hyper-heuristics has been motivated by the need for a decision criteria to solve the problem of which method is the best to solve a problem. After "No Free Lunch Theorem" [40] which claims that an algorithm never gives identical results has been published, it's observed that all search algorithms have same average performance for all problems defined on a search space. This result seems normal, since most of the possible problems do not have common structure and similar properties.

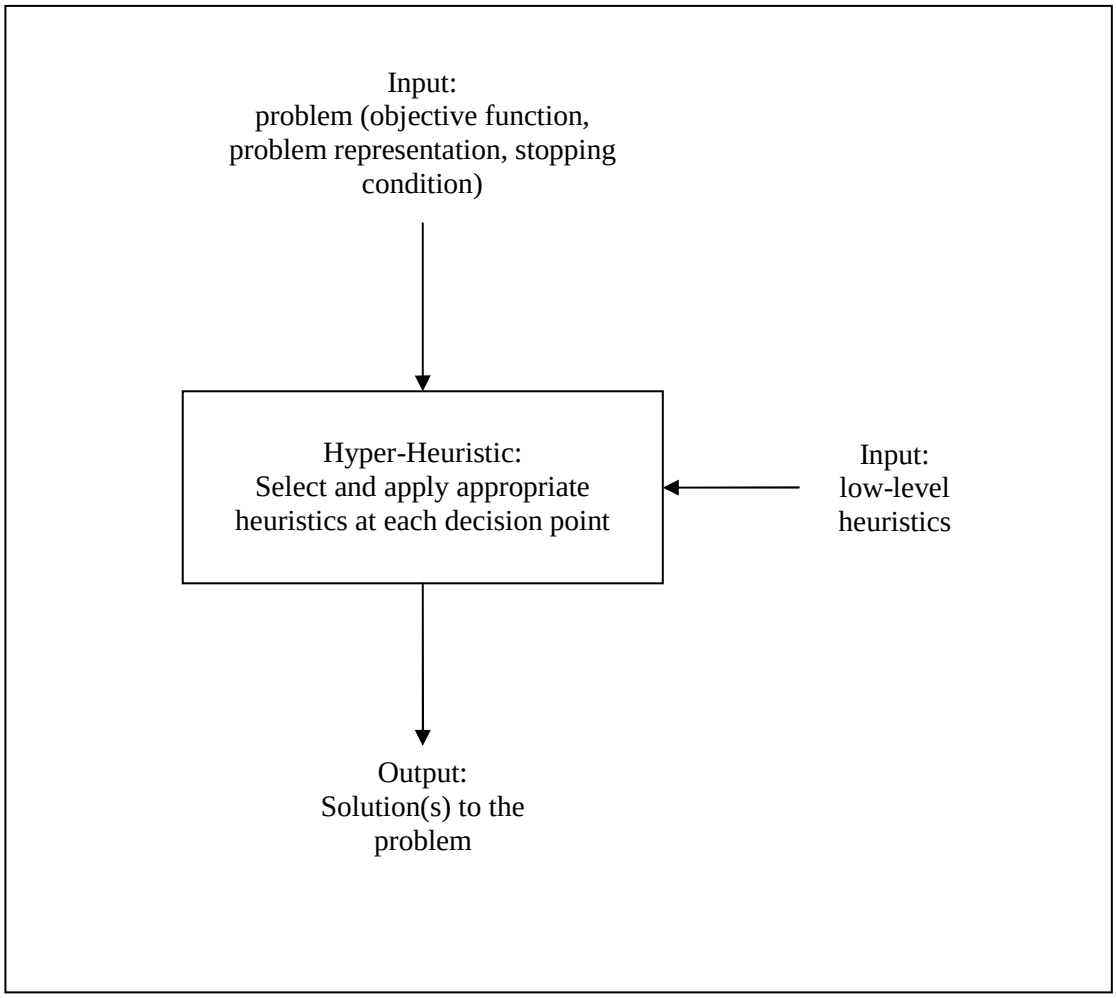



**Figure III.3.1 General hyper-heuristic framework [41]**

Therefore, the limitation of individual heuristics has been made clear. An alternative methodology which works well on different problems is required. One straightforward approach is an algorithm containing a switch statement for all problems and applying the best known heuristic for each. There are also smarter approaches which use more sophisticated techniques such as local search procedure.

Hyper-heuristics use a set of heuristics to transform the state of the problem. It associates each heuristic with the problem conditions and applies different heuristics to the different parts or states of the problem by choosing the most appropriate ones. In this manner, HH provides automatic selection of parameters (heuristics) works well and fast on a set of different problems and is applicable to as many problems as possible.

```
1. START with a set H of heuristics which are applicable to a
problem state
2. Initialize the problem state
3. Find the most suitable heuristic for the state
4. Apply the heuristic to get a new state
5. If the problem is solved
     Stop
6. Otherwise
     Go to the step 3
```

**Figure III.3.2 Hyper-heuristic algorithm [39]**

A Hyper-Heuristic framework which is shown in Figure III.3.3 works at a higher level than a metaheuristic works. The framework generally does not have any information about the problem domain. It only has a set of low level heuristics but no knowledge about the function of these low level heuristics. When a hyper-heuristic algorithm has been developed, new problem domains can be adapted by replacing the set of low-level heuristics and evaluation function.

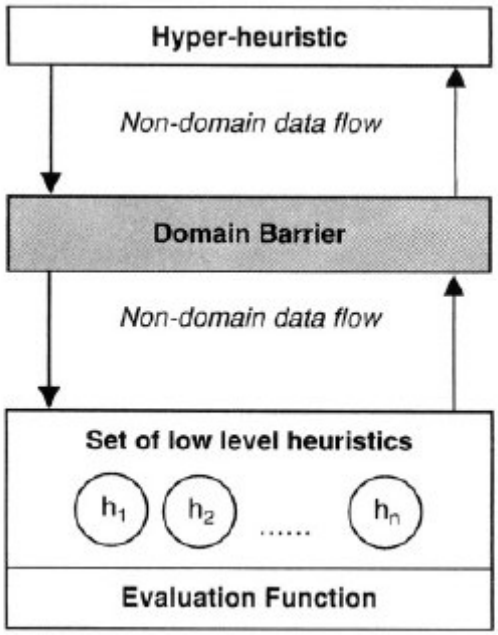



**Figure III.3.3 Hyper-heuristic Framework [39]**

Hyper-heuristic design may be considered as a three level construction, a) level 0 (problem representation), b) level 1 (low-level heuristics), and c) level 2 (hyper heuristics). In this subsection, we briefly explain each level separately.

**III.3.1.** Level 0 (Problem Representation)

If the partial solutions are dealt with, the intermediate incomplete solutions generated during the evolution phase are considered while all the solutions generated are considered in the complete solutions case.

There are three types of acceptance criteria considered in hyper-heuristic based solutions.

AM (all moves): A solution is accepted regardless of performing better or worse than the previous one.

OI (only improving): A solution is accepted if it is better than the previous one.

IE (improving or equal): A solution is accepted if it is better or same as the previous one.

It should be noted that hyper-heuristic techniques can be also considered for multiple-objective functions. Therefore, level 0 should be designed accordingly; i.e., a hyper-heuristic may select the appropriate low-level heuristic for only one objective of the problem or all objectives forming the overall objective function.

Additionally, a hyper-heuristic technique may maintain the population-based solutions or point-based solutions. Population-based solutions have parameters which take time to be tuned while point-based solutions are parameter-free and less time consuming.

**III.3.2.** Level 1 (Low-Level heuristics)

The size and the type of the low-level heuristics applied must be decided in an efficient way. The size must be selected, as small if the process is simple and as large if the hyper-heuristic works well. The type must be selected, as simple to make the implementation not difficult and as sophisticated to give efficient solutions in the problem domain. Additionally, the selected low-level heuristic may be applied once or repetitively as long as it gives better solution.

**III.3.3.** Level 2 High-level heuristic (Hyper-heuristic)

There are different learning mechanisms [44] which show how to decide which low level heuristics are selected.

**A) <u>Random approaches.</u>**

- Simple Random (SR): It randomly selects a low-level heuristic at each iteration until termination condition is satisfied.

- Random Descent (RD): It selects a low-level heuristic at random and applies it until there is no improvement in the solution.
- Random Permutation (RP): It selects a random permutation of low-level heuristics and this permutation is cycled until termination condition is satisfied.
- Random Permutation Descent (RPD): It selects a random permutation of low-level heuristics and this permutation is cycled until there is no improvement in the solution.

```
Select a solution randomly
repeat
  Select a low level heuristic uniformly at random
  Apply selected low level heuristic once to the current
  solution
until stopping condition is satisfied
```

**Figure III.3.4 Simple Random pseudocode**

```
Select a solution randomly
repeat
  Select a low level heuristic uniformly at random
  repeat
      Apply selected low level heuristic to the current
      solution
  until There is no improvement in the solution
until stopping condition is satisfied
```

**Figure III.3.5 Random Descent pseudocode**

```
Select a solution randomly
Select a random permutation of all low level heuristics
repeat
      Select the next low level heuristic in the sequence
      Apply selected low level heuristic once to the
      current solution
until stopping condition is satisfied
```

**Figure III.3.6 Random Permutation pseudocode**

```
Select a solution randomly
Select a random permutation of all low level heuristics
repeat
    Select the next low level heuristic in the sequence
    repeat
        Apply selected low level heuristic to the current
        solution
    until
        There is no improvement in the solution
until stopping condition is satisfied
```

**Figure III.3.7 Random Permutation Descent pseudocode**

**B) <u>Greedy Approach (GR):</u>** At each iteration, it evaluates the change in the objective function in the solution by applying each low-level heuristic. And the best low-level heuristic is applied so long as there is any improvement.

```
Select a solution randomly
repeat
    for all low level heuristics
        Calculate objective function improvements on the
        current solution
    endfor
    repeat
        Apply the low level heuristic giving the best
        improvement
    until there is no improvement
until stopping condition is satisfied
```

**Figure III.3.8 Greedy Approach pseudocode**

**C) <u>Choice Function Method</u>**

In addition to the simple hyper-heuristics considered above, we also use the choice function-based technique [44]. It calculates the choice function of each heuristic by considering the recent effectiveness of the heuristic, the recent effectiveness of consecutive pairs of heuristics and the amount of time the heuristic

was last called. Individual performance of the heuristic $N_j$ is defined with the equation given below [44]:

$$f_1(N_j) = \sum_n \alpha^{n-1} \frac{I_n(N_j)}{T_n(N_j)} \tag{IV.19}$$

where $I_n(N_j)$ is the change in the fitness function the $n^{th}$ last time heuristic $N_j$ was called, $T_n(N_j)$ is the amount of time taken the $n^{th}$ last time heuristic $N_j$ was called, $\alpha$ is a parameter between 0 and 1 which shows the importance of the recent performance.

After calling the $N_j$, the new value of the $f_1(N_j)$ is calculated by using the old value of the $f_1(N_j)$ by using the following formula:

$$f_1^{new}(N_j) = \frac{I_1(N_j)}{T_1(N_j)} + \alpha f_1^{old}(N_j) \tag{IV.20}$$

Aggregate performance of the heuristic $N_j$ and consecutive heuristic $N_k$ is defined as:

$$f_2(N_j, N_k) = \sum_n \beta^{n-1} \frac{I_n(N_j, N_k)}{T_n(N_j, N_k)} \tag{IV.21}$$

where $I_n(N_j, N_k)$ is change in the fitness function the $n^{th}$ last time heuristic $N_k$ was called after heuristic $N_j$, $\beta$ is a parameter between 0 and 1 which shows the importance of recent performance.

After calling $N_k$ just after $N_j$, the new value of the $f_2(N_j, N_k)$ is calculated by using the old value of $f_2(N_j, N_k)$ by using the following formula:

$$f_2^{new}(N_j, N_k) = \frac{I_1(N_j, N_k)}{T_1(N_j, N_k)} + \beta f_2^{old}(N_j, N_k) \tag{IV.22}$$

Amount of time elapsed since the heuristic $N_j$ was last called is defined as:

$$f_3(N_j) = \tau(N_j) \tag{IV.23}$$

After defining the elements of the choice function, the resulting choice function used can be defined as:

$$f(N_j)=\alpha f_1(N_j)+\beta f_2(N_j,N_k)+\delta f_3(N_j) \quad \text{(IV.24)}$$

This choice function provides both exploitation by $f_1$ and $f_2$, and exploration by $f_3$. We consider three different approaches [44] for the choice function.

- Straight Choice: It selects the low-level heuristic which gives the best value of the choice function.
- Ranked Choice: It ranks the low-level heuristics according to the choice function values and evaluates the change in the objective function by applying a proportion of the highest ranked heuristics. And the best low-level heuristics is applied.
- Roulette Choice: At each iteration, it selects a low-level heuristic Ni with probability which is proportional to $F(N_i)/\sum F(N_i)$.

The pseudocodes of the three variants of choice function are given below.

```
Select a solution randomly
repeat
    for all low level heuristics
        Calculate choice function
    endfor
    Apply the low level heuristic which has the highest
    choice function value
until stopping condition is satisfied
```

**Figure III.3.9 Straight Choice**

```
Select a solution randomly
repeat
    for all low level heuristics
        Calculate choice function
    endfor
    for n low level heuristics which have the highest choice
    function values
        Calculate objective function improvements on the
        current solution change for the highest
    endfor
    Apply the low level heuristic giving the best improvement
until stopping condition is satisfied
```

**Figure III.3.10 Ranked Choice**

```
Select a solution randomly
repeat
    for all low level heuristics
        Calculate choice function
    endfor
    for all low level heuristics
        Calculate the probability which is the proportional
  to  f(N_i) / Σ_i f(N_i)
    endfor
   Apply the low level heuristic which has the highest
   probability
until stopping condition is satisfied
```

**Figure III.3.11 Roulette Choice**

# PART IV

## PROPOSED FRAMEWORK

In this chapter, we present representation of terrains, path representation for UAVs and various objectives considered in our study, which is followed by the details of nature-inspired solutions for the path planning problem of UAVs.

### IV.1. TERRAIN REPRESENTATION

The UAV, for which the flight is planned, flies in a 3-D environment over a terrain. This terrain information is represented by heightmap structures. In computer graphics, a heightmap is a raster image used in 3-D computer graphics to store values. Figure IV.1.1 shows an example terrain which can be represented with a heightmap on a regularly-spaced grid.

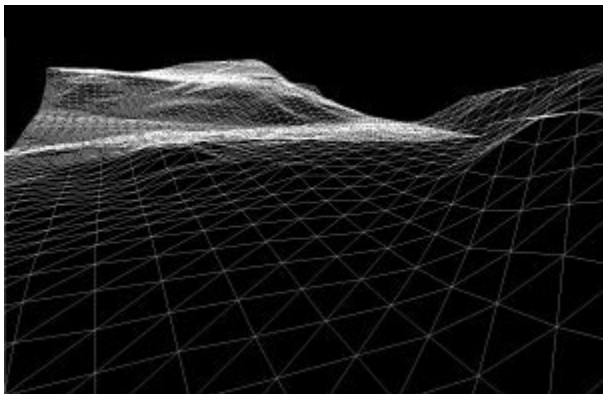


**Figure IV.1.1 Heightmap represented by regularly-spaced grids [47]**

Heightmap is generally represented by a two-dimensional matrix of which dimensions represent coordinate axis, mostly X and Z. The value of the matrix's entry gives the height value, Y coordinate, of the corresponding point. In Figure IV.1.2, the table on the left shows the cells of the height map and the values at each cell; the one on the right is a wire frame view of the terrain generated by that map.

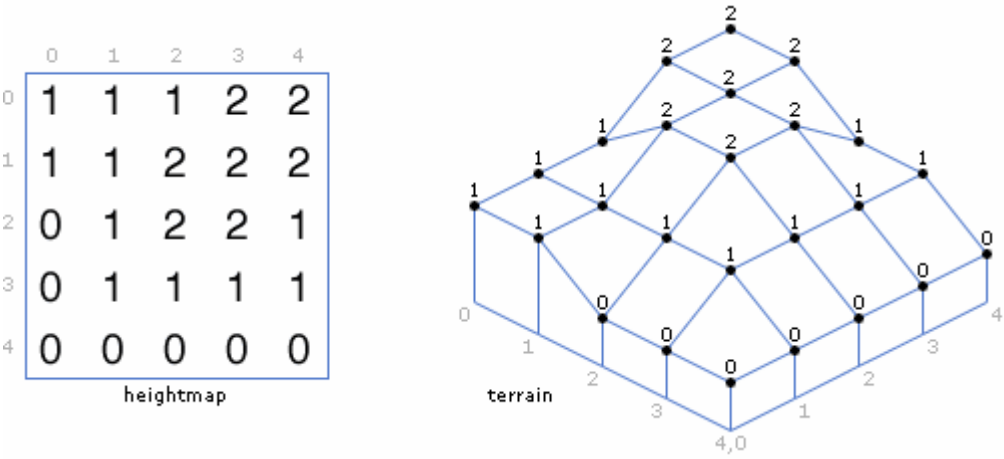



**Figure IV.1.2 The cells of the height map & view of the terrain generated by that map [48]**

Another way to picture a heightmap is to think of it as a grayscale image. By assigning a color to each height value, a height map may be displayed as an image. High points in the terrain (large values) are represented by white, and low points (small values) are represented by black:

The scale of the matrix which gives the point intervals may be changed by setting appropriate values. The height value of a point may be found via this formula:

$$height = heightmap[(column + row * size)] \qquad \text{(IV.1)}$$

where *column* and *row* are dimensions of the heightmap matrix and *size* is the size of the matrix. The height values of the points which are not stored in the heightmap matrix due to the scale may be found with linear interpolation by using the following equation:

$$height = height_1 + \frac{desired_point - point_1}{point_2 - point_1} * (height_2 - height_1) \qquad \text{(IV.2)}$$

where *desired_ point* is the point of which height value to be calculated but not stored in the heightmap, $point_1$ and $point_2$ are the closest points to the desired point stored in the heightmap matrix, $height_1$ and $height_2$ are the height values of the $point_1$ and $point_2$ respectively.

**IV.2. PATH REPRESENTATION**

In our study, the flight path which the UAV would follow is represented by B-spline curves. Since it is not efficient way to represent the path line of a UAV by straight line segments as in mobile robot applications [49] or simple splines as in [50], the B-Spline curves are selected for the path representation.

B-spline curves are piecewise polynomial curves that are differentiable to a specified order [2]. These are parametric curves constructed by calculating basis functions. Mathematically, given $n+1$ control points $P_0, P_1, ..., P_n$ and a knot vector $U = \{u_0, u_1, ..., u_m\}$, the B-Spline curve of degree k defined by these control points and knot vector $U$ is:

$$C(u) = \sum_{i=0}^{n} N_{i,k}(u) * P_i \tag{IV.3}$$

where $N_{i,k}(u)$ 's are B-Spline basis functions of degree k.

With coordinates $(x_0, y_0, z_0), ..., (x_n, y_n, z_n)$, the coordinates of the B-spline can be written as:

$$X(t) = \sum_{i=0}^{n} N_{i,k}(u) * x_i \tag{IV.4}$$

$$Y(t) = \sum_{i=0}^{n} N_{i,k}(u) * y_i \tag{IV.5}$$

$$Z(t) = \sum_{i=0}^{n} N_{i,k}(u) * z_i \tag{IV.6}$$

The B-Spline basis functions are used as weights like being in Bezier curves. The function domain is divided by knots and the basis function values have always non-zero values. The functions are defined recursively in terms of a set of knot values. If the knots are equally spaced, the knot vector or the knot sequence is said uniform; otherwise, it is non-uniform. We may define with the most common form being the uniform non-periodic one as [2]:

$$\begin{aligned} Knot(i) &= 0, && if \quad i < k \\ Knot(i) &= i - k + 1, && if \quad k \le i \le n \\ Knot(i) &= n - k + 2, && if \quad n < i \end{aligned} \tag{IV.7}$$

The knot points divide a B-Spline curve into curve segments, each of which is defined on a knot span.

The basis functions $N_{i,k}$ 's are defined recursively (algorithm known as de Boor recursion formula), using the knot values given by the following equation:

$$N_{i,1}(t) = \begin{cases} 1, & if \quad Knot(i) \le t \le Knot(i+1) \\ 1, & if \quad \begin{cases} Knot(i) \le t \le Knot(i+1) \\ and \\ t = n - k + 2 \end{cases} \\ 0, & otherwise \end{cases} \tag{IV.8}$$

$$N_{i,k}(t) = \frac{(t - Knot(i)) \times B_{i,k-1}(t)}{Knot(i+k-1) - Knot(i)} + \frac{(Knot(i+k) - t) \times B_{i+1,k-1}(t)}{Knot(i+k) - Knot(i+1)} \tag{IV.9}$$

The parameter of the curve, t, changes between 0 and n-k+2 to provide the curve points. According to the desired sensitivity of the curve, number of curve points may be adapted by dividing this interval into equal pieces.

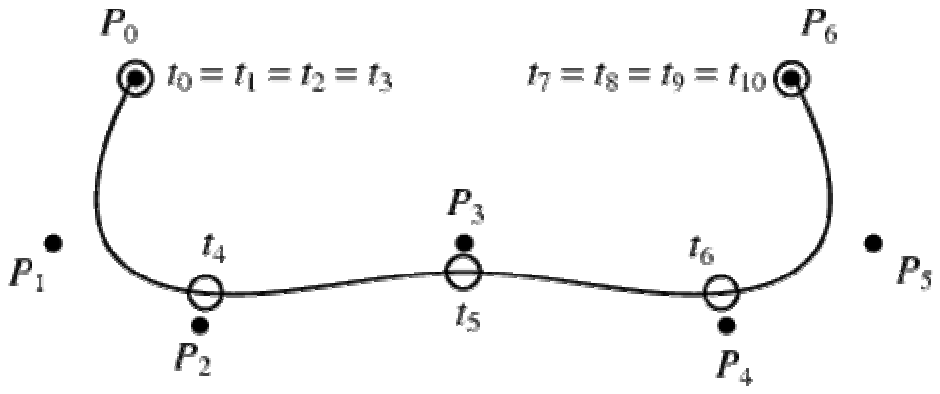



**Figure IV.2.1 B-spline curve representation [51]**

B-Spline curves are suitable to represent the path of the UAVs since they need few variables to define complicated curve paths. Moreover, a change/update in one

control point changes only the part of the curve segments that are near the changing control point, which is due to its local effect.

### IV.3. OBJECTIVES of the PATH PLANNING PROBLEM

In our study, the problem of determining the navigation path of an UAV can be considered as a multi-objective problem with which various constraints are needs to be considered. This problem is formulated as a minimization problem, which takes into account six general constraints that are listed in the following table:

| |
|---|
| C1. Constructing feasible paths without terrain collision, |
| C2. Constructing a path within the desired minimum and maximum distance to the terrain, |
| C3. Constructing a path which provides UAV to maneuver with an angle greater than minimum radius of curvature, |
| C4. Avoiding threat zones of different shapes, if any |
| C5. Minimizing the length of the path, |
| C6. Providing a flight envelope in a given line of sight |

**Table IV.3-1 Constraints in the Path Planning Problem**

These constraints can be classified into two categories: hard constraints (or technological constraints) that must be satisfied at the generated flight path, and soft constraints that are evaluated in terms of the given measure for the quality of the solution. The first four constraints listed above are hard constraints and the remaining ones are the soft constraints.

In our study, we consider the first four entries (given above) as the constraints for path planning problem; the last two entries can be considered as objectives (single or multiple objectives) for different flight missions

The quality of the constructed path by considering the above constraints of an UAV is measured by a cost function value calculated as the sum of six different cost terms:

$$f = \sum_{j=1}^{6} w_j * f_j \qquad \text{(IV.10)}$$

where $f_i$'s are cost function values described below and $w_i$'s are weights of each function determining priority of the functions. Note that each $f_i$ is associated with the corresponding constraint given in Table IV.3-1.

For any navigation problem considered, the first four cost terms ($f_1$ through $f_4$) of the above equation are considered with non-zero weights, since these are the hard constraints; and one of the remaining two cost terms (f5, f6) is added by setting the zero value for the weights of the remaining terms, based on the objective of the navigation.

The term $f_1$ in equation (IV.10) represents the number of curve points inside the solid boundary, which is for the first constraint given in the previous page. So, the penalty value is proportional to the number of discretized curve points located under the solid surface. All n path segments of the flight path are checked whether or not pass through the terrain by computing the distance between the curve point and the terrain using heightmap data. If this is true for a discrete point of the path line, a constant penalty is added to term $f_1$. Consequently, this term provides to construct the collision-free paths by giving high penalty values to the curves with more points inside the solid boundary. The formulation of the cost $f_1$ is as follows:

$$f_1 = \sum_{i=1}^{n} y_i, \quad where \;\; y_i = \begin{cases} 1 & if \;\; d_i \le 0, \\ 0 & \text{otherwise;} \end{cases} \qquad \text{(IV.11)}$$

where $d_i$ is the distance between the curve point and the terrain, and i is the index of curve points. Figure IV.3.1 presents the pseudocode for calculating $f_1$ term.

```
begin
  m ← num_of_curve_points_inside_terrain
  For each discretized point  n_i
      If   d(n_i) < 0  then
           m = m +1
  endfor
end
```

**Figure IV.3.1 Collision penalty calculation pseudocode**

The term $f_2$ is for the distance penalty of the curve, which is related with the second constraint given in Table IV.3-1. This term consists of two penalty values including safety distance penalty (which is for keeping a safety distance between UAV and the ground) and maximum distance penalty (which is for the maximum altitude from the ground due to technological restrictions or the maximum allowable altitude due to mission-specific constraints).

Safety distance penalty aims to provide a flight path far from specified minimum distance. The distance of each discretized curve point is checked for the safety distance; and the penalty value is calculated proportional to the difference between the actual distance and desired safety distance. The penalty value for each curve point forms the cost function value. However, to calculate the distance of the curve point for only one terrain point cannot be sufficient in 3-D environment since the curve point can be closer to the any other points than the corresponding terrain point. Therefore, for each curve point, the distance between the corresponding terrain point and its neighbor points is calculated; the maximum penalty value of the neighboring terrain point distances is considered for each curve point as the safety distance penalty value.

Maximum distance penalty provides a flight path that is in the range of specified maximum altitude of the UAV. The maximum allowable altitude value is considered as the altitude with respect to the sea level, due to the technological constraints of the UAV. If this term is ignored, the generated path might be the shortest one but that path can not be realized with respect to UAV's technical specifications. The penalty value is calculated proportional to the difference between the distances and allowed maximum altitude; i.e., the far the curve point is the larger penalty value it has. The term $f_2$ given below is represented in two separate terms, $f_{2,1}$ and $f_{2,2}$, which are formulated below:

$$f_{2,1} = \sum_{i=1}^{n} \max_{k \in N(.)} \left\{ \left( d_{\min} - d_{i,k} \right) / d_{\min} \right\} * y_i, \ \textit{where} \ y_i = \begin{cases} 1 & \textit{if} \ d_{i,k} \le d_{\min}, \\ 0 & \text{otherwise;} \end{cases} \quad \text{(IV.12)}$$

$$f_{2,2} = \sum_{i=1}^{n} \left\{ \left( d_i - d_{\max} \right) / d_{\max} \right\} * y_i, \textit{where} \ y_i = \begin{cases} 1 & \textit{if} \ d_i > d_{\max} \\ 0 & \textit{otherwise} \end{cases} \quad \text{(IV.13)}$$

$$f_2 = f_{2,1} + f_{2,2} \quad \text{(IV.14)}$$

where $N(.)$ is the neighbor set of the specific terrain point, $d_{i,k}$ is the distance between the curve point and the terrain point, $d_{\min}$ is the minimum safety distance from the terrain, $d_i$ is the distance between the curve point and sea level which is obtained by getting the minimum value of the heightmap and $d_{\max}$ is and maximum allowable distance from the sea level.

```
begin
  p ← safety_distance_penalty
  m ← allowable_minimum_distance
  For each discretized point  ni
      If d(ni) > m  then
          p = 0
      Else If   d(ni) < m  then
          neighbor ← number_of_neigbor_points
          pn ← penalty_of_each_point
          neighbor = d(ni)
          For each neighbor point
              pn = (m - d(ni)) / m
              p = p + pn
          endfor
 endfor
end
```

**Figure IV.3.2 Safety distance penalty pseudocode**

```
begin
  p ← max_distance_penalty
  m ← allowable_maximum_distance
  For each discretized point  ni
      If d(ni) > m  then
          p = p + (d(ni) - m) / m
  endfor
end
```

**Figure IV.3.3 Max distance penalty pseudocode**

The term $f_3$ indicates the minimum radius of curvature penalty of the path, which is for the third technological constraint given in Table IV.3-1. It is designed to prevent the UAV from exceeding the lateral and vertical acceleration limits, since the flight envelope determines the maximum radius of turns for the UAV. It's aimed to achieve a flight path with a specified minimum curvature angle by this term. To obtain penalty value, angle between two line segments of the curve is calculated for each three neighbor curve points as in shown in Figure IV.3.4.

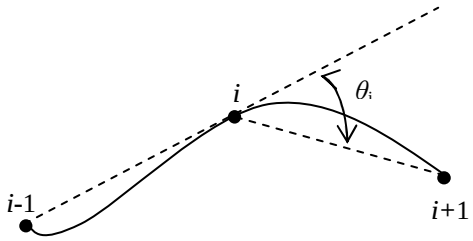



**Figure IV.3.4 Schematic representation of the curvature angle**

As in previous terms, penalty value is added proportional to the difference between the angle and specified minimum angle. The term $f_3$ can be formulated as the following:

$$f_3 = \sum_{i=1}^{n} \{(\Theta - \theta_i)/\Theta\} * y_i, \qquad where \;\; y_i = \begin{cases} 1 & if \;\; \theta_i \le \Theta, \\ 0 & \text{otherwise;} \end{cases} \tag{IV.15}$$

where $\theta_i$ is the angle between the curve point and the neighbors, and $\Theta$ is the minimum curvature angle.

```
begin
  p ← curvature_radius_penalty
  m ← allowable_minimum_curvature_radius
  For each discretized point n_i
       np1 and np2 ← neighbor points to point n_i
       v1 ← vector between np1 and n_i
       v2 ← vector between np2 and n_i
       Θ ← angle between v1 and v2
       If Θ > m  then
            p = p + (Θ - m) / m
  endfor
end
```

**Figure IV.3.5 Curvature radius penalty pseudocode**

The term $f_4$ indicates the number of discretized curve points which is located in the specified threat zone. The area which has dangerous threats may be defined by geometric shapes such as squares, rectangles, triangles or circles.

```
begin
  p ← number_of_curve_points_inside_threat_zone
  A ← threat_zone
  For each discretized point n_i
       If n_i inside A
            p = p + 1
  endfor
end
```

**Figure IV.3.6 Threat zone penalty calculation pseudocode**

The term $f_5$ is the length of the flight path calculated by adding the distance of two neighbor curve points for each discretized point. This term is used in order to minimize the flight path lengths and can be formulated as:

$$f_5 = \sum_{i=0}^{n} \left[ (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2 \right]^{1/2} \quad \text{(IV.16)}$$

```
begin
  l ← length_of_the_curve
  For each discretized point n_i
      d ← distance between the n_i and n_i+1
      l = l + d
  endfor
end
```

**Figure IV.3.7 Shortest path calculation pseudocode**

The term $f_6$ indicates the number of discretized curve points which are not located in the line of sight of the specified point. The line of sight states the visibility of the path from the point. In our study, we have considered the line of sight between the source point of the terrain and the curve points which represent the path. This penalty value is evaluated for the line of sight of each curve point as shown in Figure IV.3.8.

```
begin
  p ← number_of_curve_points_not_in_los
  For each discretized point n_i
      x ← x coordinate of the n_i
      y ← y coordinate of the n_i
      z ← z coordinate of the n_i
      If x >= z then
          For j = 1 to x
              test_x = j
              test_y = (test_x * y) / x
              test_z = (test_x * z) / x
              p ← point test_x, test_y, test_z
              If d(p) <= 0 then
                  p = p + 1
                  continue with other point
          endfor
      Else
          For j = 1 to z
              test_x = (test_x * x) / z
              test_y = (test_x * y) / x
              test_z = j
              p ← point test_x, test_y, test_z
              If d(p) <= 0 then
                  p = p + 1
                  continue with other point
          endfor

  endfor
end
```

**Figure IV.3.8 LOS penalty calculation pseudocode**

In order to evaluate cost value given in Equation (IV.10), all $f_i$'s are normalized by using equation (IV.17) to get values in the same range after their values are calculated. This normalization transforms $f_i$ values into the range of [0;1].

$$normalized_value = \frac{value - value^{\min}}{value^{\max} - value^{\min}} \qquad \text{(IV.17)}$$

where $value$ is the real value of the function, $value_{\min}$ is the minimum value which the function can take and $value_{\max}$ is similarly the maximum value which the function can take. Since the objective is to minimize the cost function values, the lower values of cost function $f$ in (IV.10) means the higher performance of the individual.

The weights ($w_i$s) given in the main equation can be experimentally determined or can be set according to different aspects such as terrain specifications, mission specific goal definitions (i.e., flight envelope in a given line of sight) etc. If the main objective is to obtain feasible flight paths, weights may be determined in such a way that the corresponding term (one of the $f_i$ s) dominates the rest. In our experimental study, equal weights are considered.

## IV.4. GA-BASED FORMULATION of the PROBLEM

### **IV.4.1.** Representation of individuals

In our study, each individual which is a path of the UAV is represented by B-Spline curve control points. B-Spline curve is a parametric curve which is described by control points. Each chromosome has control points which have x, y and z coordinates with floating point values in 3-D environment case as shown in Figure IV.4.1.

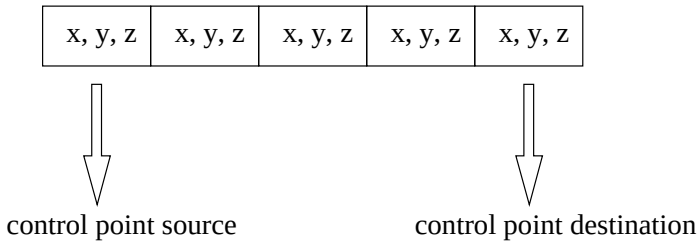



**Figure IV.4.1 Individual Representation by B-Spline Curve**

The first and the last points of the individual represent source and destination points, respectively. These two points are fixed for all individuals; the other free-to-move control points, which are located in ordered manner in the chromosome, may take different values. The number of control points may differ in different

populations or in different individuals in a same population. In Figure IV.4.2, two paths, one has 5 control points and another has 7 control points are shown in an artificially generated terrain.

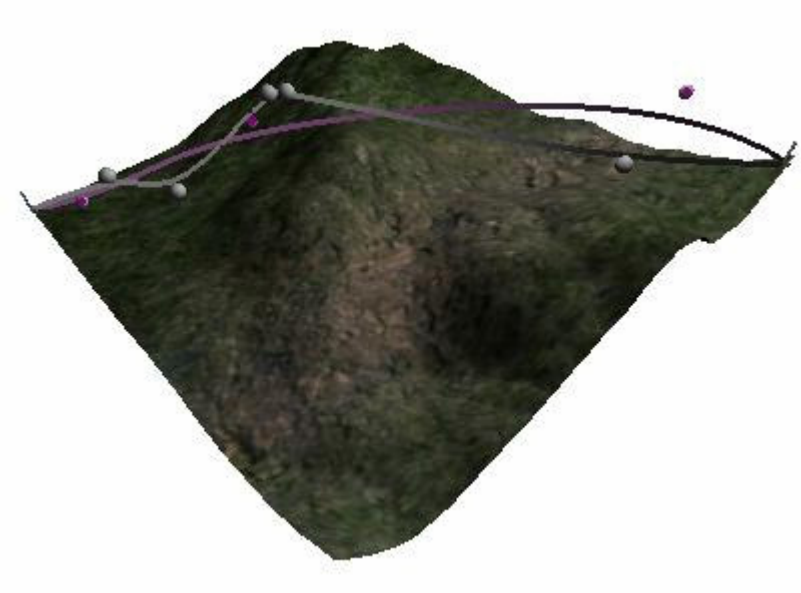


**Figure IV.4.2 The Paths with Different Number of Control Points**

**IV.4.2.** Initial Population

Initial population of the evolutionary algorithm in our problem is generated randomly by considering the boundaries of the terrain. The x and z coordinates are selected randomly from the range $[0..TerrainSize-1]$; and the y coordinate of the point, which will draw the height of the individual, is set randomly by considering the feasibility in that case. To be able to help the feasibility of the curve, the y coordinate of the control points are generated in upper side of the terrain. If a non-feasible point which collides the terrain is created, its y coordinate is re-generated by random number generator until it provides feasibility of the point. This does not guarantee the collision-free B-Spline curve since its curve points may collide with the terrain although its control points are feasible; but, this method increases the probability of feasibility of the curves.

**IV.4.3.** Parent Selection Mechanism

The selection of individuals for mating is carried out by tournament selection method [3]. This selection process has two stages:

- Select a number of k individuals randomly
- Select the parent with the best fitness among selected k tournaments

where k is the tournament size, which is a variable parameter of the Genetic Algorithm phase. It should be noted that these two stages are repeated for two individuals, which is followed by the crossover operator.

### **IV.4.4.** Survivor Selection Mechanism

In our study, a steady-state population model is considered with an elitism-based survivor selection mechanism. At each iteration or generation, crossover and mutation operators are applied on selected individuals in order to generate two offsprings. Then, the best offspring among them is selected and it is put in place of the worst individual in the population.

The stopping criterion of our experiments is to reach a predefined number of iterations.

### **IV.4.5.** Crossover and Mutation Operators

#### **IV.4.5.1.** Crossover Operators

Two crossover operators are considered in our study. One is simple *One-point crossover* which simply cuts the parents from one point randomly, and splits both parents at this same point; then it creates two new individuals by exchanging the portions left.

The other crossover operator is a *knowledge-based crossover operator* that aims to shorten the length of the curve by considering the locations of the crossover point. This operator is applied after deciding the appropriate crossover points for two individuals as shown in Figure IV.4.3. If using the selected crossover point in the parent causes to construct a curve segment that turns back the direction to the starting point of the path, then, this crossover point is skipped; and it is shifted to the next crossover point in the same parent (i.e. the next control point) until it finds an appropriate one. Same operation is applied for each offspring. Therefore, the crossover point may be different for two mates. The knowledge-based crossover operator is especially helpful for mates having different number of points, due to the distinct distribution of the control point locations.

```
begin
  cp1 ← crossover_point_of_parent1
  cp2 ← crossover_point_of_parent2
  cp ← randomly selected crossover point
  d1 ← distance between the crossover point of parent1
and destination point
  d2 ← distance between the crossover point of parent2
and destination point
  while d1 > d2
       cp1 = cp1 + 1
  endwhile
  while d2 > d1
       cp2 = cp2 + 1
  endwhile
  apply crossover to the parents
end
```

**Figure IV.4.3 Knowledge-based crossover pseudocode**

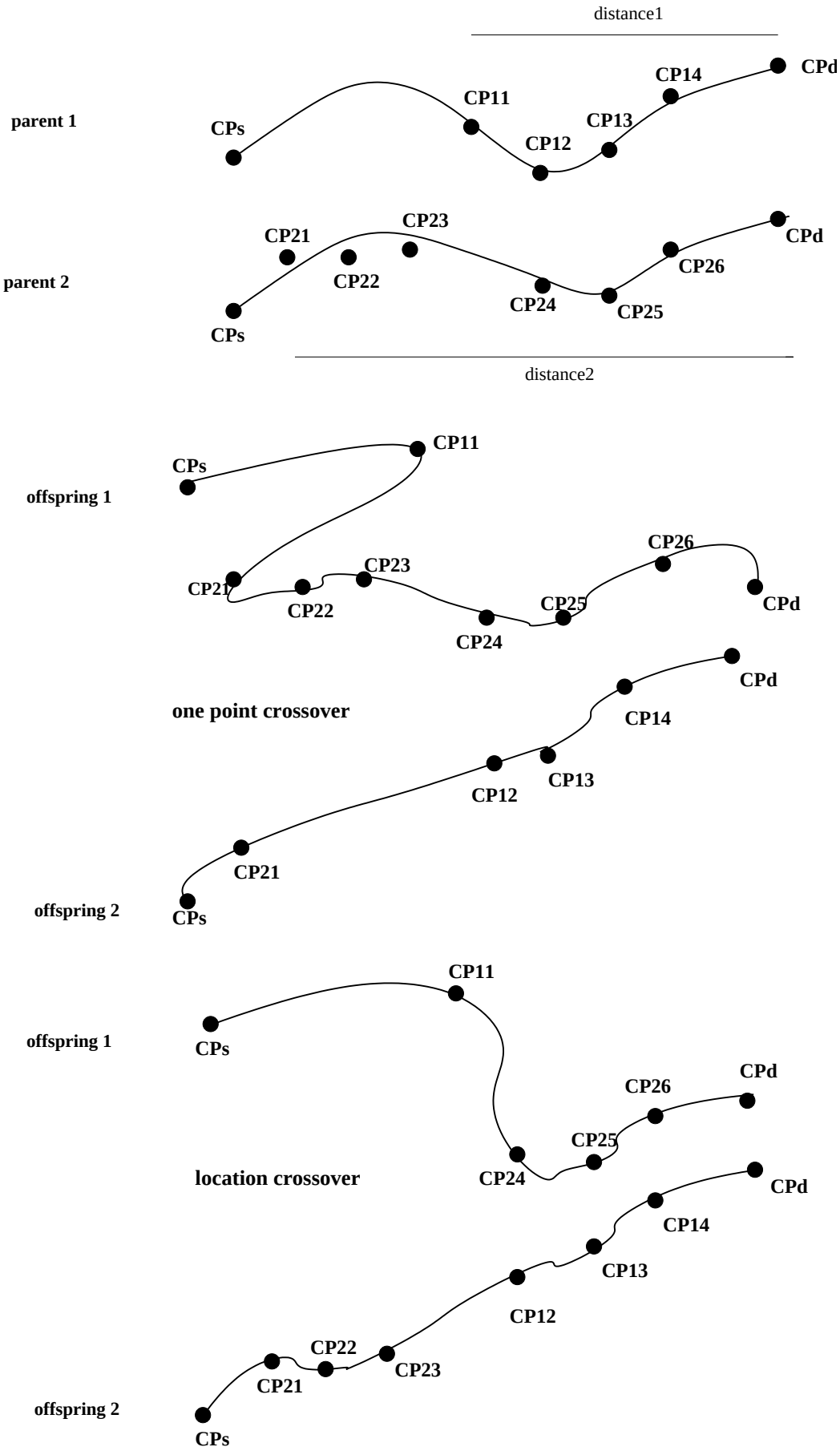



**Figure IV.4.4 Crossover Operators**

**IV.4.5.2.** Mutation Operators

Knowledge based mutation operators which work on the free to move control points are proposed as part of our study. Our mutation operators alter the number or the position of the control points according to the fitness values of the control points. As mentioned before, a B-spline curve is divided into segments, which are the separated parts between the control points. In general, the operator starts working by finding the worst segment, which is the one that has the minimum fitness (maximum cost) of the given curve. It should be noted that while calculating the fitness of the curve segments, only the first four terms in the cost function (given in Equation (IV.10)) is considered; i.e., the technological or hard constraints are considered.

We consider three different mutation operators in our experiments, which are update, insertion and deletion operators.

**Update:** This mutation operator updates the worst control point of the individual. The quality of each control point is computed by the aggregate cost of the curve points on the two neighbor curve segments; and it should be noted that the cost of a segment includes only the first four terms of the objective. For instance, the cost of the control point 2 is calculated by adding the cost of the segment 1 and segment 2, where segment 1 is the piece of the B-Spline curve which is between the control points one and two, and the segment 2 is the piece of the B-Spline curve which is between the control points two and three. After the control point which has the maximum cost value is found, this control point is replaced by a new one in order to increase the fitness of the curve. In order to achieve this objective, the new point is created in a feasible region. The x and z coordinates of the new point are set by considering the following equations:

$$x_k^* = x_{k-1} + (x_{k+1} - x_{k-1}) * k + \varepsilon$$
$$z_k^* = z_{k-1} + (z_{k+1} - z_{k-1}) * k + \varepsilon \qquad \text{(IV.18)}$$

where $k$ and $\varepsilon$ are terrain-specific parameters. $\varepsilon$ is assigned randomly between 0 and 1, k is taken as 0.5. The $y$ coordinate of the new control point is randomly set from the feasible region generated by using the new $x$ and $z$ values. The $y$ coordinate is set by adding a displacement to the current value of $y$ so that final value will be within the minimum and maximum safety distances.

```
begin
  cp ← control_point_with_the_lowest_fitness
  ncp ← new_control_point
  ε ← randomly_selected_parameter
  ncp_x = cp(i)_x + cp(i-1)_x – cp(i+1)_x * k + ε
  ncp_z = cp(i)_z + cp(i-1)_z – cp(i+1)_z * k + ε
  select randomly feasible y coordinate ncp_y
  update the control point cp with ncp
end
```

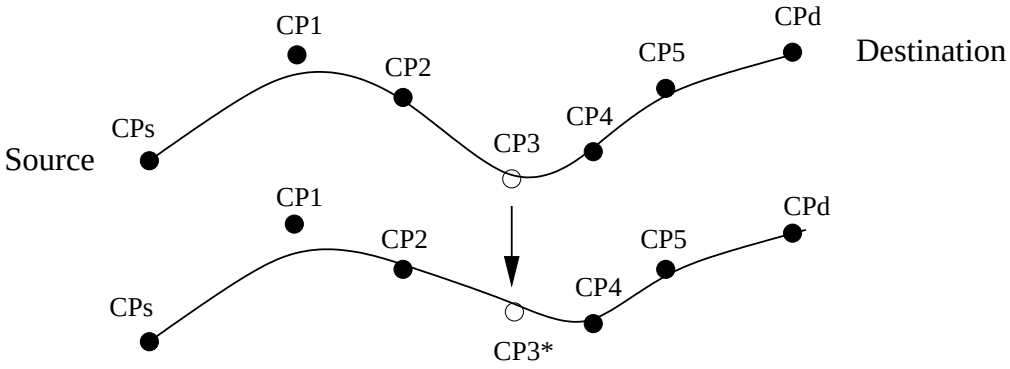
**Figure IV.4.5 Knowledge-based update mutation pseudocode**

**Insertion:** This mutation operator increases the number of the control points in a given individual and targets to increase the quality of the individual. After getting the fitness values of each curve segment, the segment which has the lowest value is established. Although update operator considers aggregate values of segments in order to update a control point, the insertion operator considers individual segments in order to insert a new control point. In this operator, first the worst curve segment (according to cost that is computed as in the update operator) is determined, and then the curve point on this segment which has the maximum cost is found. By using the x and z coordinates of the selected curve point, a new control point of the B-Spline curve is created dynamically in a feasible region by considering the safety distance constraints.

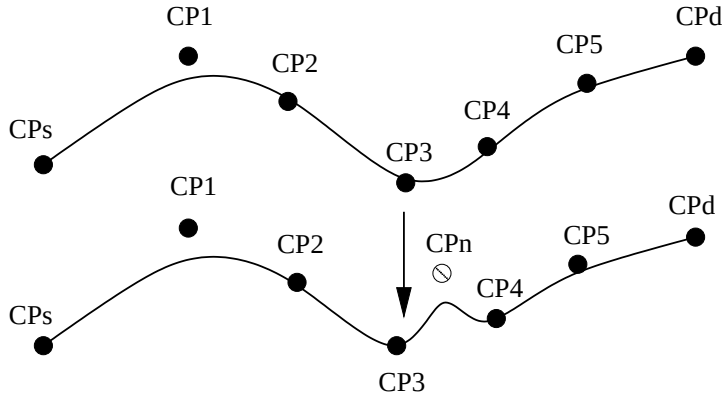
```
begin
  find the worst segment on the B-Spline curve
  find the worst curve point cp on the segment
  ncp ← new_control_point
  ncp_x = cp_x
  ncp_z = cp_z
  select randomly feasible y coordinate ncp_y
  insert the control point cp
end
```

**Figure IV.4.6 Knowledge-based insert mutation pseudocode**

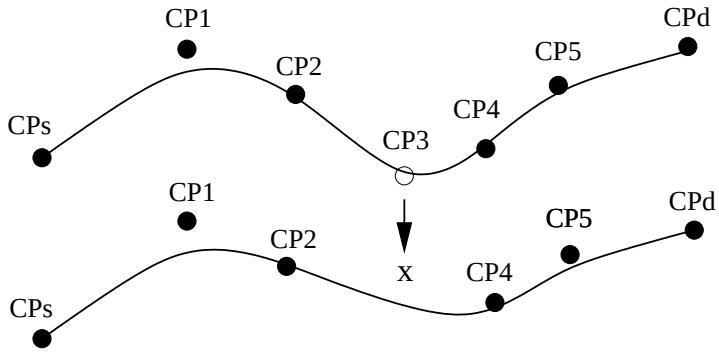
**Deletion:** This operator also alters the number of the control points in the individual. It finds the worst control point in the B-Spline curve (by considering aggregate fitness values of segments as in the update operator) and simply deletes this point from the curve. Therefore, this individual has one less control point number. Figure IV.4.7 shows how to apply the three mutation operators on a given B-Spline curve.




a. Mutation Operator 1: UPDATE




b. Mutation Operator 2: INSERTION




c. Mutation Operator 3: DELETION

**Figure IV.4.7 The types of Mutation Operators**

## IV.5. PARTICLE SWARM OPTIMIZATION

In this section, we describe the PSO implementation details for the UAV path planning problem.

The elements of the PSO method are the particles and the position, velocity and adaptability of these particles in the swarm which contains the particles representing the solution for the problem. In our case, the particles are represented by the paths formed by B-spline curves. The position of the particle $(P)$ is the B-spline curve having the control points while the velocity of the particle $(V)$ used to update the position is the position change of the control points of the curve. The adaptability of the particle $(F(P))$ is the fitness value which determines the best position of each particle and which particle has the best value in the swarm.

```
begin
  initialize swarm with random particles
  initialize each particle
  do
       for each particle
            calculate the best fitness value obtained so
            far   (Pbest)
       endfor
  find the best particle in the swarm (Gbest)
  for each particle
       update velocity
       update position
       update fitness value
  endfor
  while the number of run is reached
end
```

**Figure IV.5.1General scheme for PSO Algorithm as pseudocode**

The velocity of the particle namely the B-spline curve is represented by the change of 3-D coordinates of each control point of the curve. Figure IV.5.1 presents the pseudo code for velocity calculation in PSO algorithm

```
for each curve
  find random r1 and r2 values
  v ← velocity_of_the_particle
  pBest ← control_point_of_pBest
  gBest ← control_point_of_gBest
  for each control point of the curve ni
       t1 = vix
       t2 = c1 * r1 * (pBestix - nix)
       t3 = c2 * r2 * (gBestix - nix)
       vix = t1 + t2 + t3
       t1 = viy
       t2 = c1 * r1 * (pBestiy - niy)
       t3 = c2 * r2 * (gBestiy - niy)
       viy = t1 + t2 + t3
       t1 = viz
       t2 = c1 * r1 * (pBestiz - niz)
       t3 = c2 * r2 * (gBestiz - niz)
       viz = t1 + t2 + t3
  endfor
endfor
```

**Figure IV.5.2 Velocity calculation in PSO Algorithm for path planning problem**

Moreover, the position change of the B-spline curve is obtained by adding the control point of the curve (current position) and the velocity values belonging to each control point (Figure IV.5.3).

```
for each curve
  v ← velocity_of_the_particle
  for each control point of the curve ni
       nix = vix
       niy = viy
       niz = viz
  endfor
endfor
```

**Figure IV.5.3 Velocity vs control point location**

Particle Swarm Optimization requires a set the parameters which are explained in detail at Part V of this thesis. Generally, we set these parameters with the values in the literature.

## IV.6. HYPER-HEURISTICS

In this section, we explain the design and implementation details of the Hyper-heuristic method for the UAV path planning problem. We first present problem representation part and low-level heuristics considered, which is followed by the design of high-level heuristics.

### **IV.6.1.** Problem Representation

In our study, the hyper-heuristic works on complete solutions during its execution, that is, the hyper-heuristic is to find a complete path for the UAV. It constructs a complete initial solution and tries to improve the solution by using low-level heuristics.

Three different acceptance criteria is used in our study including AM (All Moves), OI (Only Improving), IE (Improving or Equal). We apply each of these methods for different runs in order to measure their effect on the quality of solutions.

While all cost function elements are considered when exploring the search space, point-based hyper-heuristic is used due to its simple implementation which does not have many parameters to be decided.

### **IV.6.2.** Low-level Heuristics

For the UAV path planning problem, we described different mutation mechanisms which are applied to the current solution. Once a solution is created randomly in a feasible region, the low-level heuristic (mutation type) which is chosen by the learning mechanism is applied to the individual. Then according to the acceptance criteria, the solution is updated with the mutated individual. This process continues until the stopping criteria (i.e., the number of runs) is satisfied.

```
begin
  decide on the learning mechanism and acceptance criteria
  select a solution
  repeat
      choose low-level heuristic by the learning
      mechanism
      apply the low-level heuristic on the solution
      accept solution by using the acceptance criteria
  until the number of iterations reached
end
```

**Figure IV.6.1 Hyper-heuristic pseudocode**

The low-level heuristics used by the hyper-heuristic method are the combination of the mutation types mentioned in the GA part and a set of simpler ones. They can be categorized as random and knowledge-based mutation mechanisms, which are explained below in detail.

**R1 - Random update:** In this low-level heuristic, one of the control points in the B-spline curve representing the individual is selected and it's replaced by a randomly created point in the terrain.

**R2 - Random insert:** In this low-level heuristic, a randomly created point in the terrain is appended to the B-spline curve as a new control point. This mechanism increases the number of control points at each iteration.

**R3 - Random delete:** In this low-level heuristic, one of the control points in the curve is chosen randomly as in the update mutation and this control point is removed from the individual by decreasing the number of control points of the B-spline curve.

**R4 - Random insert-delete:** In order to be able to keep the number of control points in the individual in a reasonable range, one of the random insert or delete mutation types which are selected randomly applied to the individual.

**R5 - Random insert-delete-update:** In this case, one of the random insert, update or delete mutation types are selected randomly and applied to the individual.

**R6 - Random threshold:** In this case, minimum and maximum threshold values are used to decide the mutation type to be applied. If the number of control points is less than specified minimum threshold value, then random insert mutation is applied. If it is more than specified maximum threshold value, then, random delete mutation is applied to decrease the number of control points to the desired range. Otherwise, one

of the insert, delete or update mutation types are selected to apply. Thus, the number of control points of the individuals in the population remains between the specified minimum and the maximum values.

**K1 - Knowledge-based update:** In this one, the worst control point of the B-spline curve is replaced a new one which is created by the way explained in the GA part. This operator does not work randomly for selecting the control point to remove as well as creating the control point to locate instead of the removed one.

**K2 - Knowledge-based insert:** In this case, after the worst segment of the B-spline curve is established by the method described in the GA part a new control point is inserted to the B-spline curve. The new control point is formed by considering safety distance constraints as in the update mutation.

**K3 - Knowledge-based delete:** This low-level heuristic selects the worst control point by considering the fitness value as in the update operator, and deleted this point from the curve.

**K4 - Knowledge-based insert-delete:** One of the knowledge-based insert or delete mutation is selected randomly as in the random insert-delete mutation mechanism described above.

**K5 - Knowledge-based insert-delete-update:** It selects the mutation to be applied among the knowledge-based insert, delete and update mutation types.

**K6 - Knowledge-based threshold:** This one also behaves like random threshold mutation mechanism. It selects delete knowledge-based insert mutation if the individual's number of control points is less than the minimum value specified, however it selects knowledge-based delete if the number of control points is more than the maximum value. Otherwise, if the number of control points is between the minimum and maximum threshold values the mutation type is selected randomly among three knowledge-based mutation operators including update, insert or delete.

### **IV.6.3.** Hyper-Heuristic Techniques

In our study, we consider a set of simple hyper-heuristics and choice function as learning mechanism. The hyper-heuristics considered in our study are random methods (including the Simple Random (SR), the Random Descent (RD), the Random Permutation Descent (RPD), a greedy approach and the choice function [44]. The details of these techniques were presented at PART III of this thesis.

# PART V

## EXPERIMENTAL STUDY

In this section, we'll describe the computational experiments comparing the algorithms proposed. We first present terrain generation methods and algorithms to simulate 3-D environment. The experimental study consists of two parts including the pre-experimentation phase to set the algorithms' parameters and the main experimentations.

### V.1. TERRAIN GENERATION

To simulate a 3-D environment, jME (jMonkey Engine) [53], a Java scene graph API based on OpenGL, is used. For both terrain data generation and visualization, jME API methods are applied. The API is easily adapted to generate and visualize terrain since it is open source and has many features. Landscape generation phase creates height map data by different methods in different sizes. There are many terrain generator algorithms which artificially generate height map data representing synthetic terrain. Some of the well known algorithms are the Hill Algorithm [54, 55], Midpoint Displacement Algorithm [56, 57], Fault Formation Fractal Generation Algorithm [58] and Particle Deposition Algorithm [59, 60]. The terrains considered in our experimental study are created by the Hill Algorithm and the Midpoint Displacement Algorithm.

#### **V.1.1.** Hill Algorithm

It creates random circular displacements stacked on one another which finally resemble a hilly landscape. It starts with a flat terrain. After it picks a random point $C$ with coordinates $C_x, C_y, C_z$ on the terrain and a random radius size $r$, it raises a hill on the terrain centered at the point $C$, having the given radius $r$. The height of a sample point $A$ with coordinates $A_x, A_y, A_z$ which is located in the hill can be calculated as [54]:

$$A_y = r^2 - \left((A_x - C_x)^2 + (A_z - C_z)^2\right) \qquad \text{(V.1)}$$

This process repeats as many times as needed. Terrains generated by different number of iterations are shown in figure.

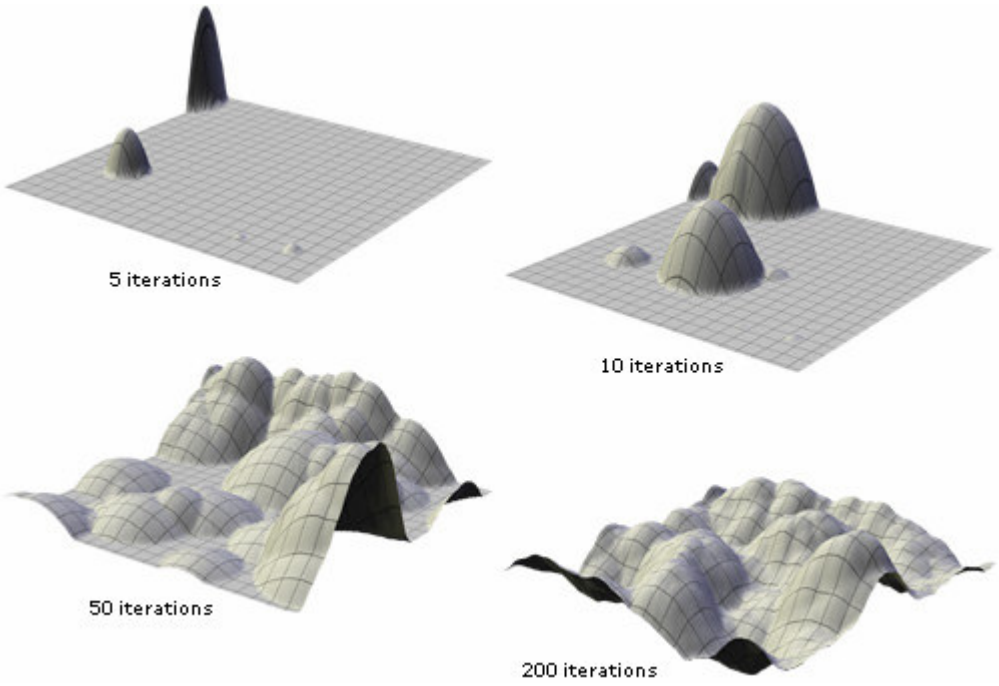



**Figure V.1.1 Multiple Hills [55]**



**Figure V.1.2 A terrain generated by Hill Algorithm**

**V.1.2.** Midpoint Displacement Algorithm

It creates terrain by displacing the center of a polygon multiple times. It subdivides triangles, squares and diamond-shaped ones [56]. One-dimensional midpoint displacement algorithm starts with a horizontal line and continues by dividing the lines from the midpoint as shown in Figure V.1.3.

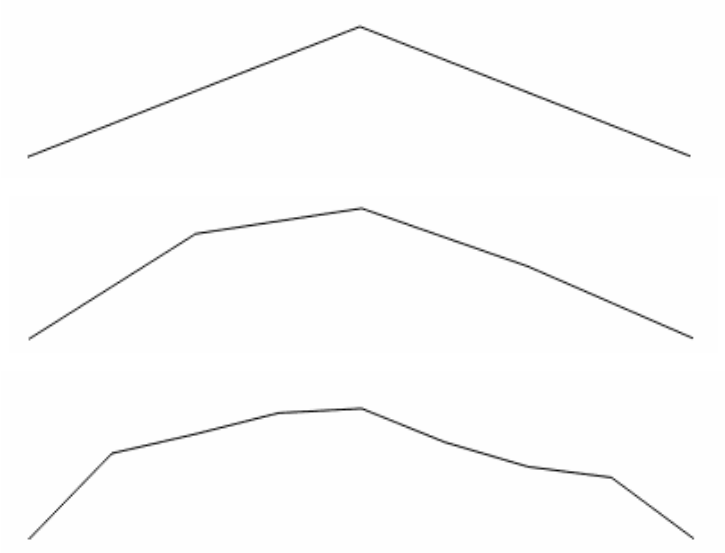


**Figure V.1.3 Iterations of displacements in one dimension [57]**

Two dimensional diamond-square algorithm starts with a plane and subdivides the plane by calculating the midpoint of the grid which is called as the diamond step and calculating the midpoint of the sides of the plane which is called as the square step.



**Figure V.1.4 A terrain generated by Midpoint displacement algorithm**

As mentioned above, terrains considered in our study are created by midpoint displacement and hill algorithms are used. To examine the behavior and performance of the heuristic methods, two terrains each has created by different parameters which build easy and more difficult structures are created by midpoint displacement algorithm and two terrains which have different hill numbers are created by hill algorithm. Mainly, 128x128 terrain size is considered by using a set of additional arguments in order to set roughness or smoothness of the terrain. The experiments have been executed for these four different terrains as shown in Figure V.1.5 where first two terrains are generated by using the Midpoint Displacement algorithm and the other two are generated by using the Hill algorithm.

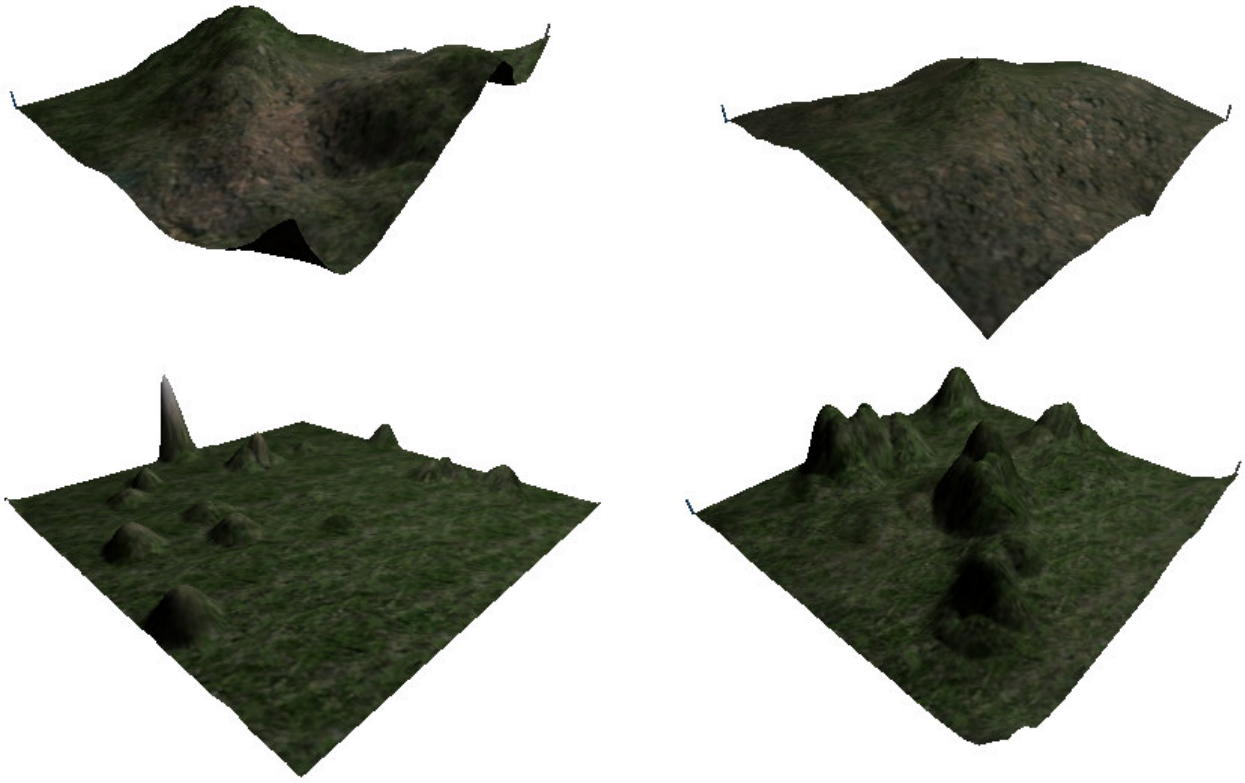


**Figure V.1.5 Sample terrains used in our experiments, above Terrain1 and Terrain2 generated by midpoint displacement, below Terrain3 and Terrain4 generated by hill algorithm**

## V.2. PRE-EXPERIMENTATION PHASE

First part of our experiments aims to select the best appropriate parameters for each heuristic method. Since the heuristic methods which we implemented for the UAV path planning problem have many parameters which need fine tuning to give the best results, this preparation phase forms the most important phase of the testing stage. In order to decide the heuristic-specific parameters, the experiments were repeated by using four different terrains, each of which was tested five times. By getting the parameters which give the best performance for each method, the experiments which compare the performance of each heuristic method including Genetic Algorithms, Particle Swarm Optimization and Hyper-heuristics are executed for different terrain structures.

### **V.2.1.** Experiments for Setting GA Parameters

There are many GA parameters which affect the performance of the algorithm. We have taken some of these parameters for pre-tests including population size, crossover rate, crossover type, mutation rate, mutation type and one problem specific parameter; number of control points. These tests are performed for five times for each four terrains by getting 1000 number of generations. After deciding the values for these parameters, the effect of the generation size in the performance advance is examined and the best value is used in the subsequent testing phases. Each parameter is tested by three level value including low value, medium value and high value as shown in Table V.2-1. Crossover and mutation types tested in this pre-experimentation phase are also listed in the Table V.2-1.

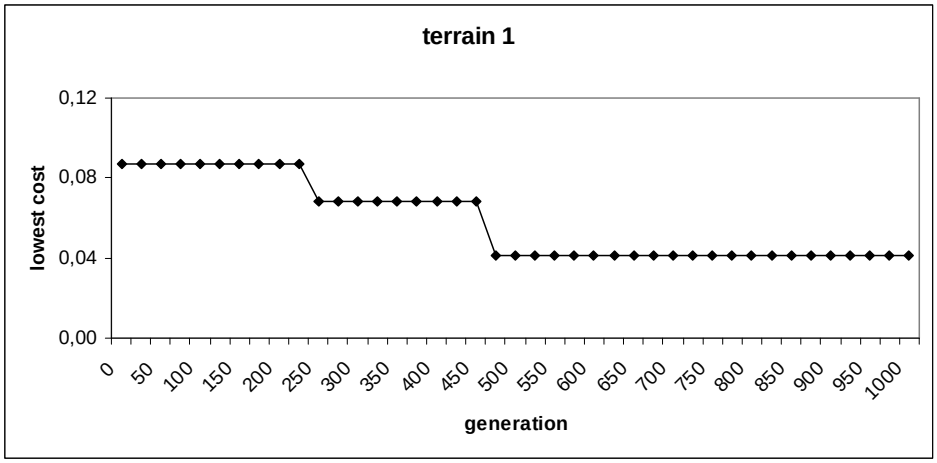
| **Parameter** | **Low value** | **Medium Value** | **High Value** |
|---|---|---|---|
| Population size | 100 | 200 | 300 |
| Tournament size | 3 | 5 | 7 |
| Crossover rate | 0.5 | 0.8 | 0.9 |
| Crossover type | T1: One-point | T2: Knowledge based | T3: T1 or T2 (random) |
| Mutation rate | 0.05 | 0.10 | 0.15 |

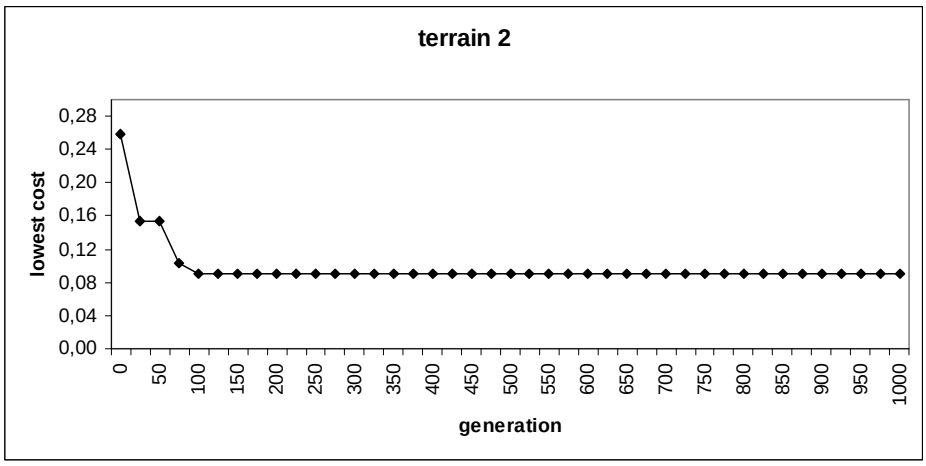
| Mutation type | T1: update | T2: Insert&delete | T3:Insert&delete (threshold) | T4: insert/delete/update |
|---|---|---|---|---|
| # of control points | 5 | 7 | 10 | 15 |

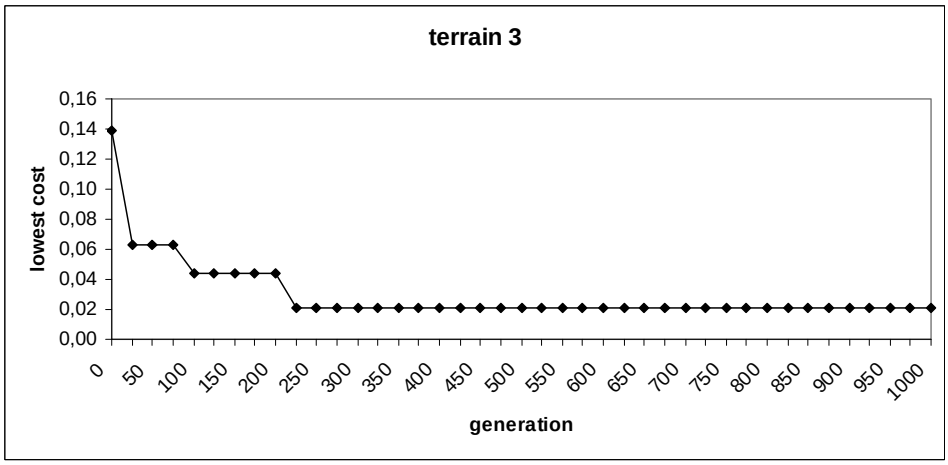
**Table V.2-1GA parameter setting**

Based on the experiments, population size is set to 200; the tournament size is set to 5. In our experiments, we consider equal values for function weights given in Equation (IV.10). The threshold values for distance and curvature radius are set according to the terrain height values. It is observed that different crossover and mutation rates give better results for different crossover and mutation types. Specifically, update mutation gives its best performance when the mutation rate is equal to 0.10; and, the random insert/delete mutation gives its best performance when mutation rate is equal to 0.15. It's also observed that the number of control points yielding the best performance varies for different terrain types. The effect of the number of control points on the quality of the solution is also measured as part of our experimental study.

The effect of the generation size on the performance is observed by getting the lowest and the average cost values. We consider one-point crossover with the rate of 0.8, and update mutation operator with the rate of 0.10 for this set of experiments. We also take different number of control points for different terrain types by using the result of the pre-experimentation phase. Specifically, a chromosome with 5 control points (including the source and destination points) gives best results for the first and third terrain (Terrain 1 and Terrain3); and solutions with 7 control points outperforms other alternatives for Terrain 2 and Terrain 4. In order to measure the performance of varying the numbers of control points, minimum number of control points is set to 5.





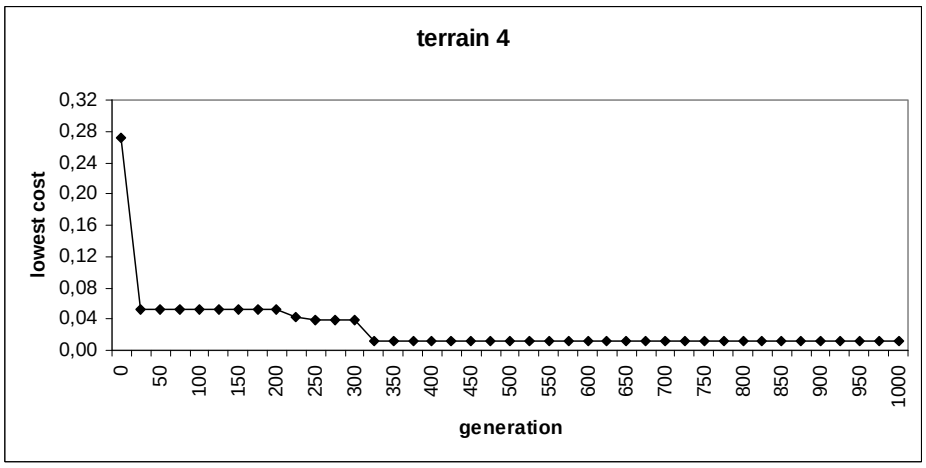
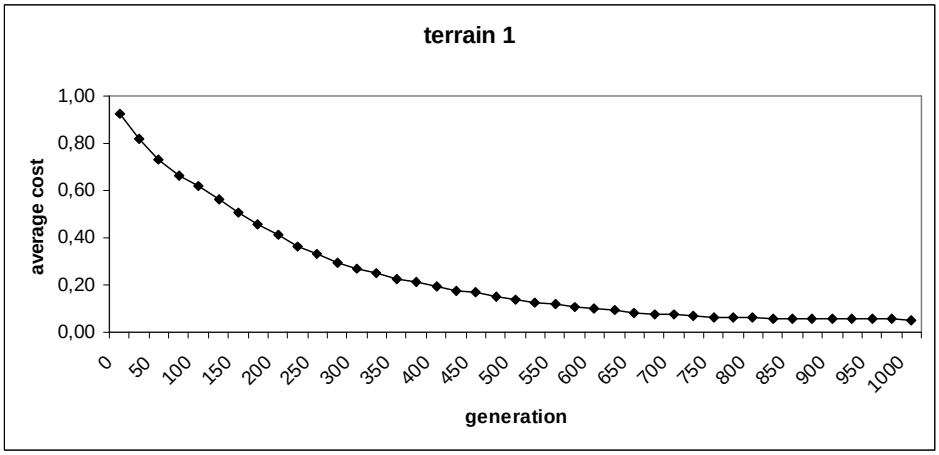
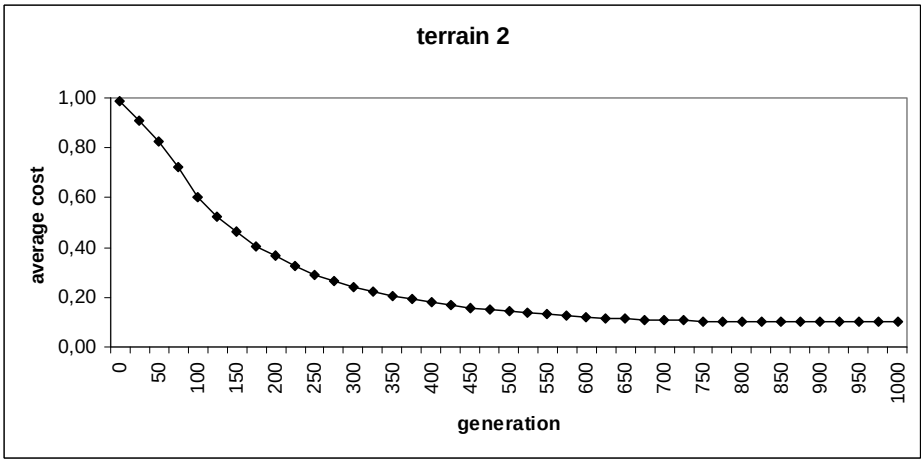



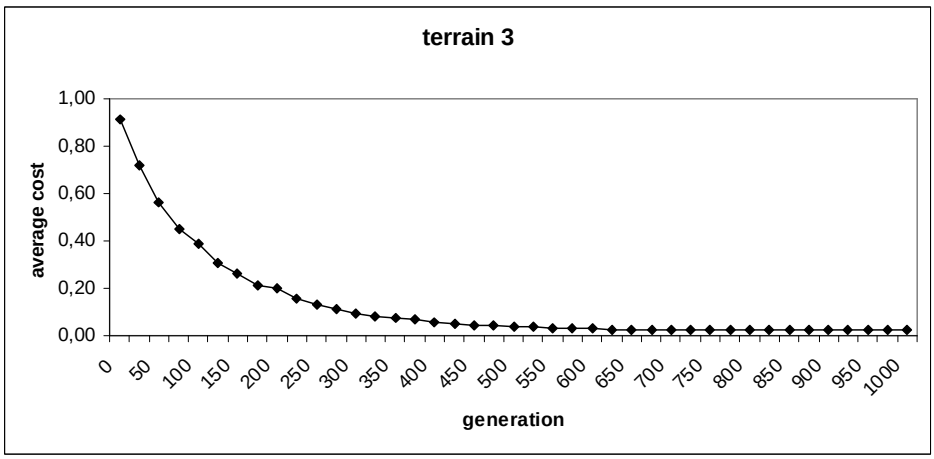

**Figure V.2.1 Lowest cost value of each generation for four terrains**

It was observed that our algorithm provides better performance with the increase in the number of generations. After a specific point, the performance can not increase as previous increments while the running time increases almost linearly. While average cost of the population continues decreasing as shown in Figure V.2.2, the lowest cost value becomes constant after a point (see Figure V.2.1). We consider

these limit values (different values for different terrains) as generation size for the later experiments.



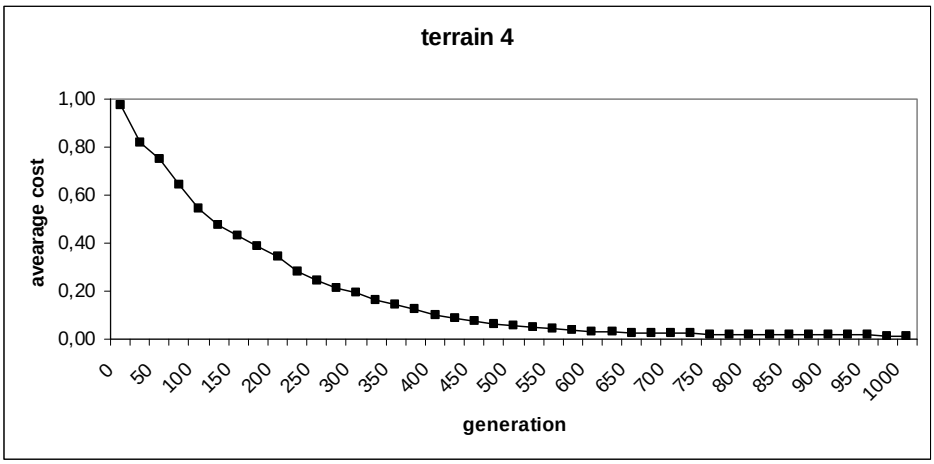
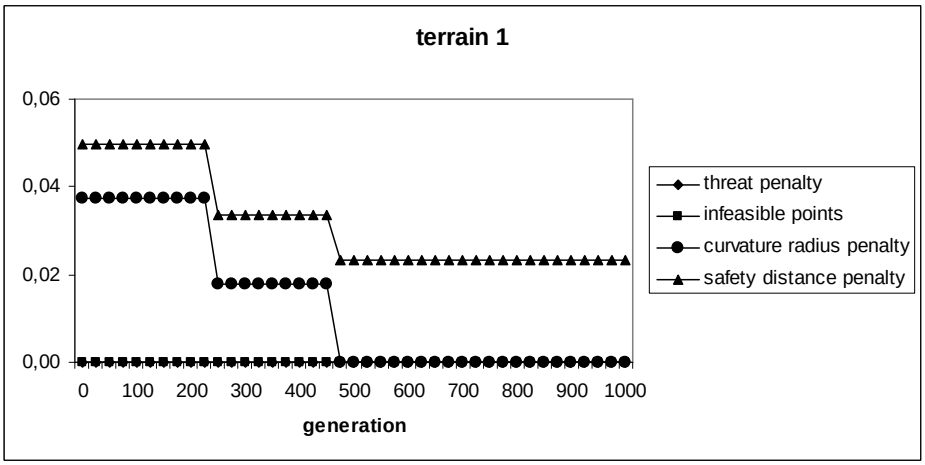



**Figure V.2.2 Average cost value of each generation for four terrains**

We measure the feasibility of the results for the four different terrains as well. Individual feasibilities of the best solution (with respect to four constraints of the given objective) are measured at each generation (see Figure Figure V.2.3). It's observed that collision penalty and threat zone penalty values are all zero beginning from the first generation; i.e. it constructs feasible paths without terrain collusion where UAV travels far from the threat zone specified. On the other hand curvature radius penalty decreases to zero value after some iteration. However, safety distance penalty (which is set when the constructed path is not within the desired minimum distance to the terrain and maximum distances to the sea level) does not have zero value during GA run. Although the safety distance penalty does not reach zero value, it decreases as generation size increases and has lower values at the end of the 1000$^{th}$ generation.



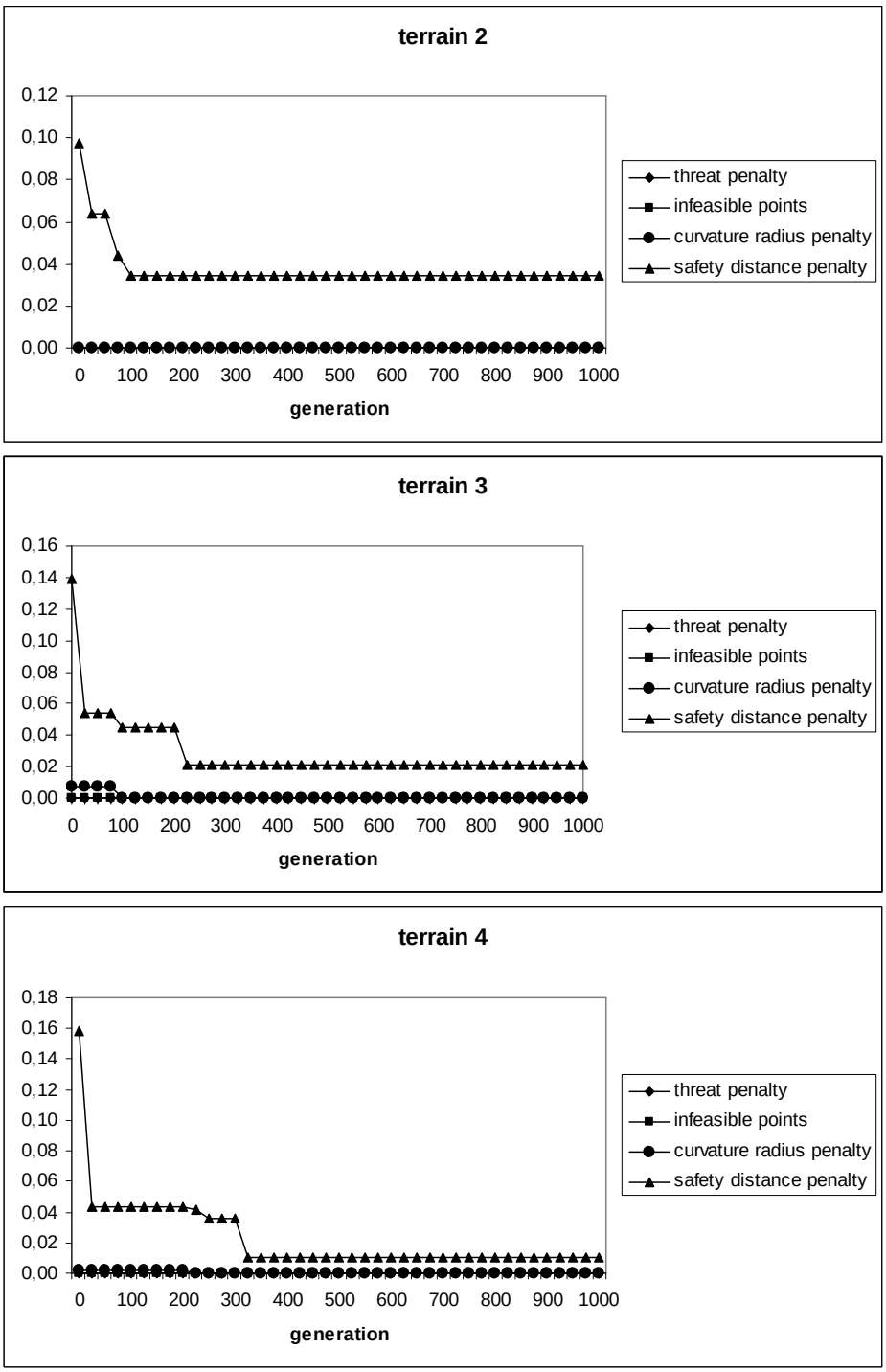



**Figure V.2.3 Feasibility of the best solution in each generation by considering four constraints of the unified objective**

The number of control points in the B-spline structure is also varied in order to measure its effect on solution quality. For easier terrains (i.e. normally one or two wide hills), smaller number of control points have higher performance; and cost value of the best individual in a population generally increases as number of control points increases. For the 128x128 size terrains, it seems that the number of control points selected for test phase is enough to observe the effect of the different number

of control points. It's obviously seen that the most successful individuals have 5 control points in case of easier terrains as shown in Figure V.2.4. For easier terrains, smaller number of control points is enough to provide better paths. As the number of control points increase, the path to route becomes more complex and especially the length of the path increases which results in worse cost values.

However, as terrain structure becomes harder (i.e there are large number of hills with multiple peaks), more control points are necessary to deal with the difficulty. Better results are observed in our experiments for larger number of control points. Generally, individuals which have 7 control points give the lowest cost in the population; while less or more control points may results in cost advance for difficult or complex terrains. For terrain 2, the effect of the number of control points on the cost value is shown in Figure V.2.5.

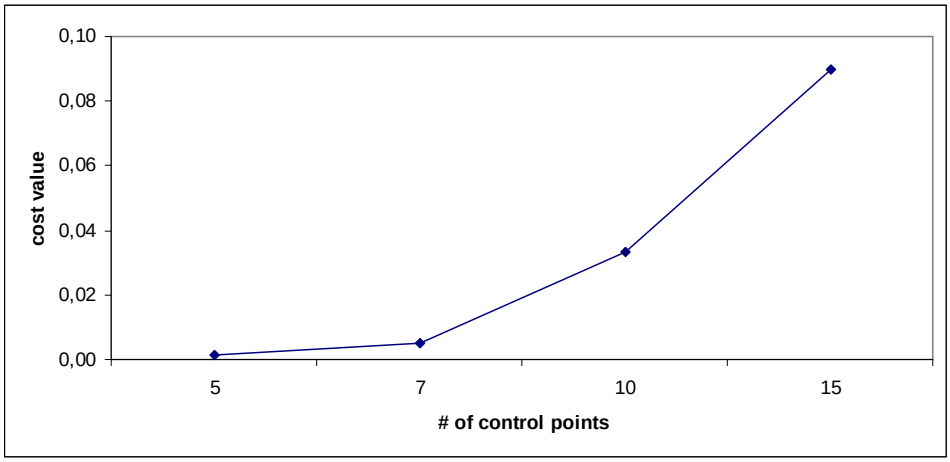



**Figure V.2.4 Cost value of the best solution by varying the number of control points for terrain3**

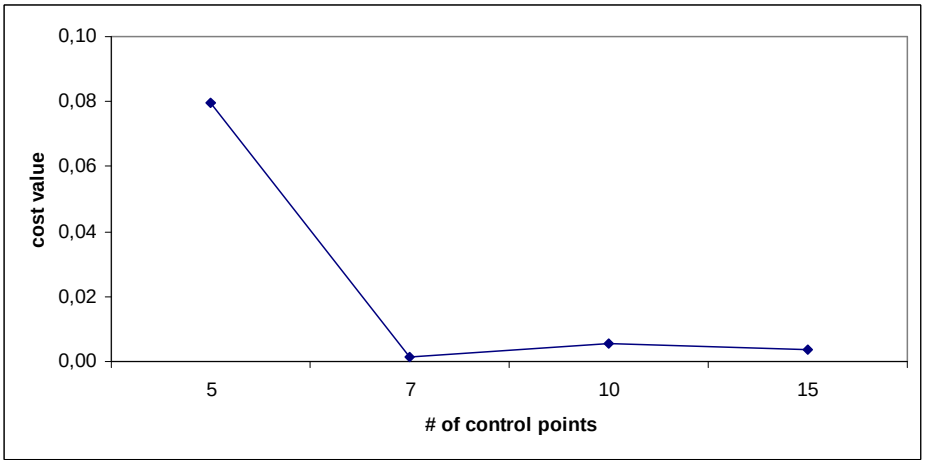



**Figure V.2.5 Cost value of the best solution by varying the number of control points for terrain2**

The performance of mutation types for different terrain structures has also been examined in our experimental study. In the pre-experimentation phase, mutation

rates are decided for different terrains. While for easier terrain structures, 0.10 mutation rate gives better performance values; 0.15 mutation ratio performs well for difficult terrain structures to deal with the constraints. Therefore, these values are used to compare the performance of the mutation types. The other parameter values are set to the values determined in the earlier tests including population size, tournament size and number of control points. As seen in Figure V.2.6, the mutation types behave in the same manner for different terrains. While the update mutation has maximum normalized cost value, the random insert/delete/update mutation gives the lowest cost values for both types of terrains. The insert/delete and threshold mutations have performance better than update mutation and worse than insert/delete/update mutation. The cost values differ in dimension for different terrain types.

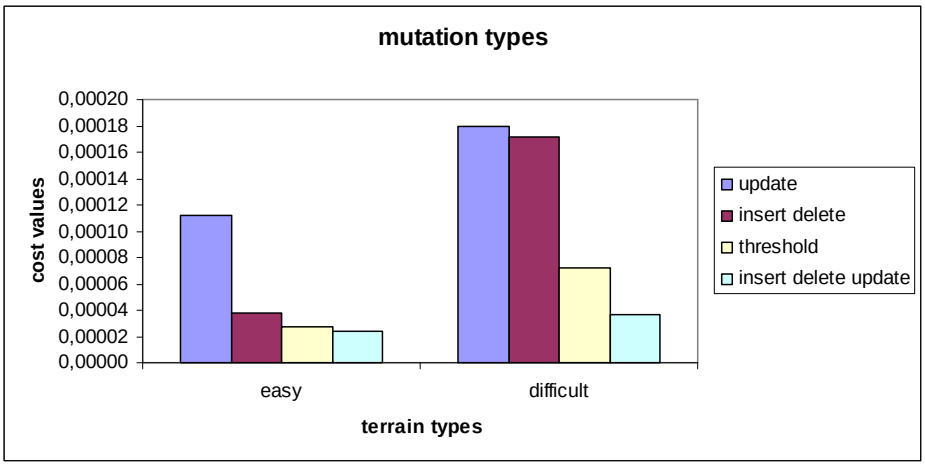



**Figure V.2.6 Performance of four different mutation operators on different terrains**

The performance of crossover types for different terrain structures has been also examined in our experimental study. In the pre-experimentation phase, crossover rates are decided for different terrains. For easier terrain structures, the best performance is observed when mutation rate is equal to 0.90; and, it decreases to 0.80 for difficult terrain structures. Therefore, these values are used to compare the performance of the crossover types. The other parameter values are set to the values determined in the earlier tests. As seen in Figure V.2.7, the crossover types show similar performance for different terrains. While the simple one-point crossover has shown the worst performance with the maximum normalized cost value, the random crossover which selects one-point or location crossover operator randomly, gives the lowest cost values for both types of terrains.

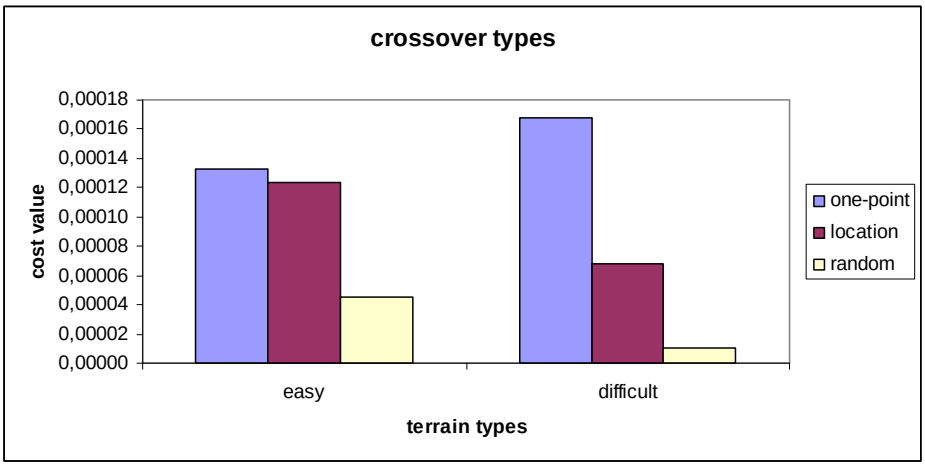



**Figure V.2.7 Performance of three different crossover operators on different terrains**

At the end of the experimental study for Genetic Algorithms, it's investigated that while some parameters give better results for different values for different terrain structures, some parameters behave in similar manners for different terrain types. In the section of the experimental study, in which the comparison of the algorithms proposed has been examined these pre-determined values as shown in

Table V.2-2 have been used for the experiment sets.

| **Parameter** | **Setting** |
|---|---|
| Population size | 200 |
| Tournament size | 5 |
| Crossover rate | 0.9 for easy terrains<br>0.8 for difficult terrains |
| Crossover type | T3: T1 or T2 (random) |
| Mutation rate | 0.10 for easy terrains<br>0.15 for difficult terrains |
| Mutation type | T4: insert/delete/update |
| # of control points | 5 for easy terrains<br>7 for difficult terrains |

**Table V.2-2 Parameter Settings for the Genetic Algorithm**

### **V.2.2.** Experiments for Setting PSO Parameters

Particle Swarm Optimization does not have many parameters as in the GA. In the pre-experimentation phase for PSO, few parameters are tested to get the best performance including the number of particles, and c1 and c2 which represents cognitive and social behavior of the particles. Inertia weight is taken as 1 in all experiments. Moreover, as in GA part, the number of control points is tested for different values.

| Parameter | Parameter Values to be Tested | | | |
|---|---|---|---|---|
| Number of particles | 20 | 30 | 40 | 50 |
| c1 cognitive parameter | 1 | 2 | 3 | 4 |
| c2 social parameter | 1 | 2 | 3 | 4 |
| # of control points | 5 | 7 | 10 | 15 |

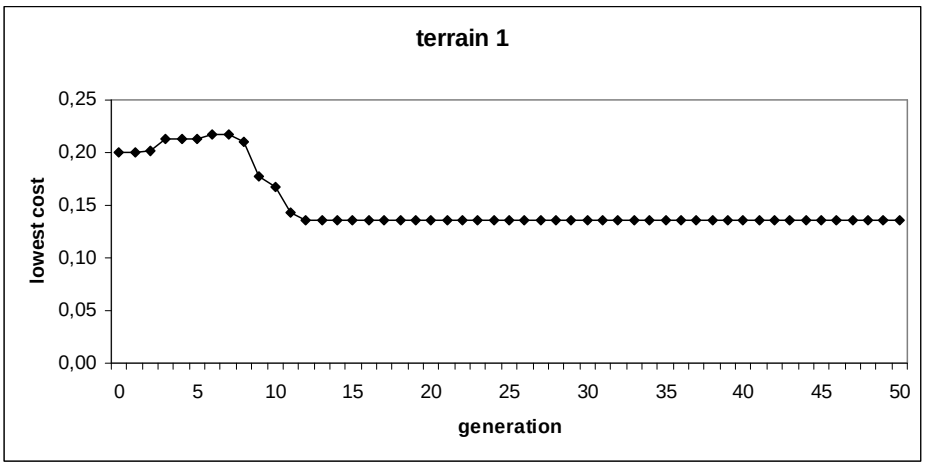
**Table V.2-3 PSO Parameter Setting**

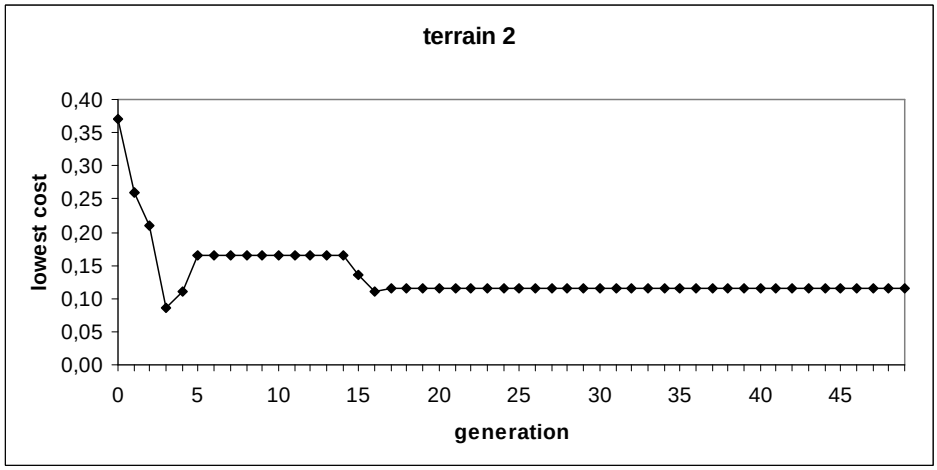
Four different artificial terrain considered in GA are also considered for PSO tests. These tests are performed for five times for each four terrains by getting 1000 number of generations. Both the lowest cost and average cost values of each experiment set are taken from the test results and the parameter values which give the lowest cost values are selected for the experimentation phase where different heuristics will be compared. The selected values for each parameter which give the best results in the experiments executed for each terrain structures are listed in Table V.2-4. The particles which have different number of control points give better performance for different terrain structures like in GA. We used 5 control points for Terrain 1 and Terrain 3, 7 control points for Terrain 2 and Terrain 4.

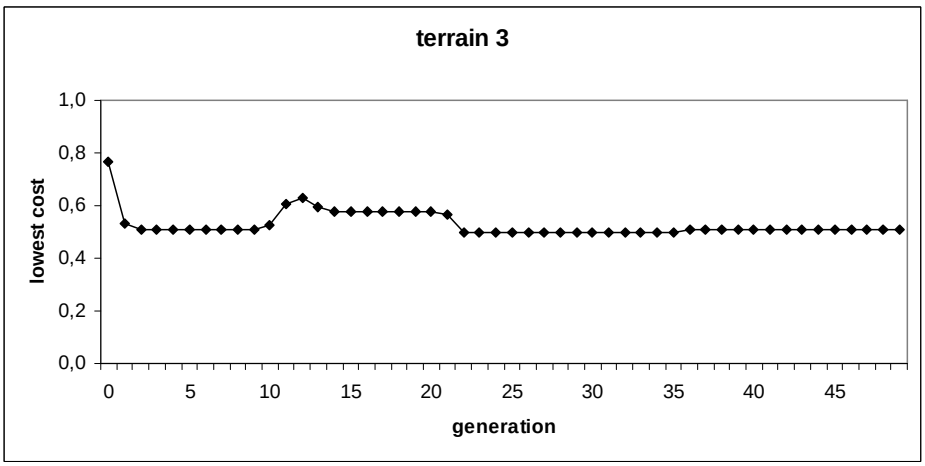
| Parameter | Setting |
|---|---|
| Number of particles | 20 |
| c1 cognitive parameter | 2 |
| c2 social parameter | 3 |
| # of control points | 5 for easy terrains<br>7 for difficult terrains |

**Table V.2-4 Final settings for the PSO Algorithm**

After setting the PSO parameters as in Table V.2-4, the lowest and average cost value of the particles during PSO run is observed for different four terrains.


terrain 1
lowest cost
0,25
0,20
0,15
0,10
0,05
0,00
0
5
10
15
20
25
30
35
40
45
50
generation


terrain 2
lowest cost
0,40
0,35
0,30
0,25
0,20
0,15
0,10
0,05
0,00
0
5
10
15
20
25
30
35
40
45
generation


terrain 3
lowest cost
1,0
0,8
0,6
0,4
0,2
0,0
0
5
10
15
20
25
30
35
40
45
generation

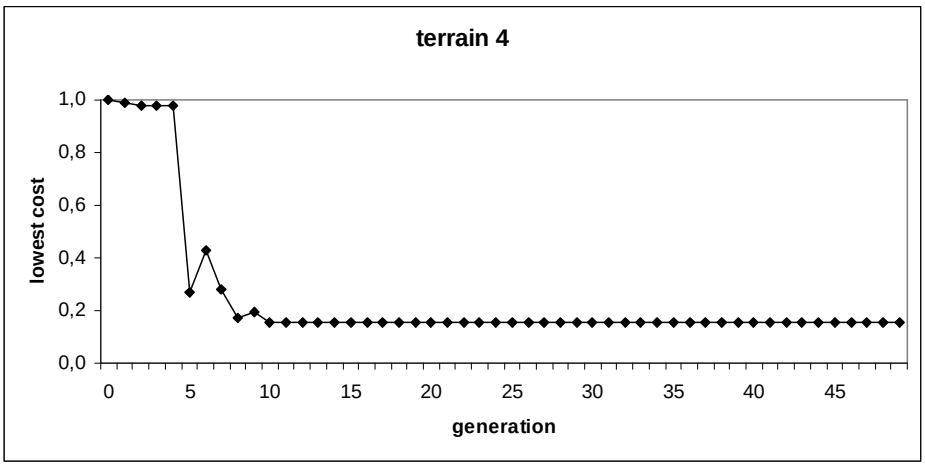
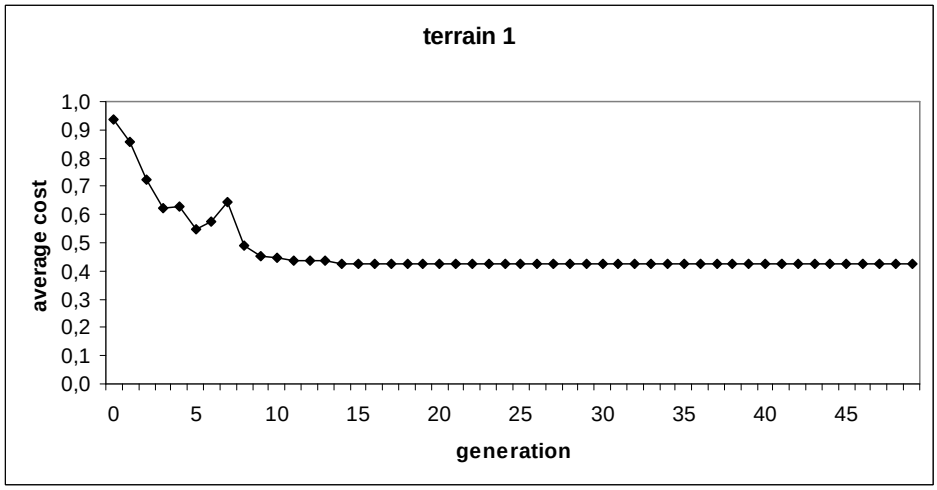
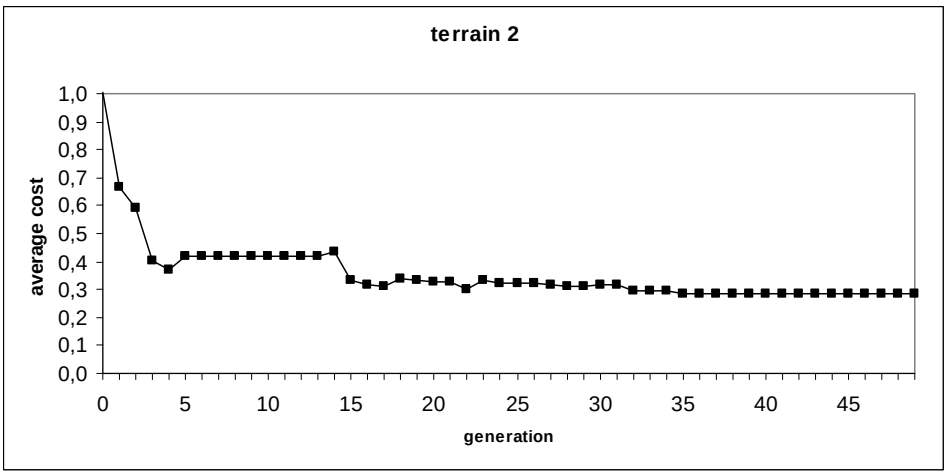
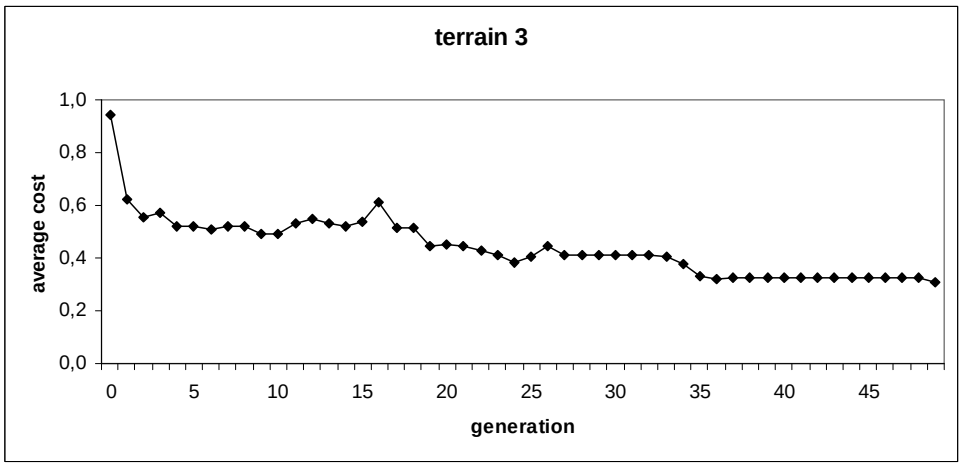



**Figure V.2.8 Lowest cost value of each generation for four terrains**

Unlike GA, the cost value of the swarm does not decrease as generation increases. Especially, the lowest cost value for different terrain types may escalate suddenly but increases at the next generation. However, the lowest cost value becomes stable at a low value satisfying the problem constraints after some number of generations. This generation number is generally lower such as 20-30. Although our experiments have been executed for 1000 generations, the cost value does not change after this low amount. Therefore, 50 generation size has been used in the later experiments phase. The average cost also has become stable after some number of generations. When the lowest cost comes to a good point which has not improved anymore, the cost of whole swarm does not improve too. Due to the swarm effect of the PSO algorithm, the improvement on the swarm stops when the improvement on the best particle stops. If the lowest cost in Figure V.2.8 and average cost in Figure V.2.9 is examined, it can be seen that the decline in the average cost of the swarm goes in parallel with the decline in the lowest cost of the swarm.





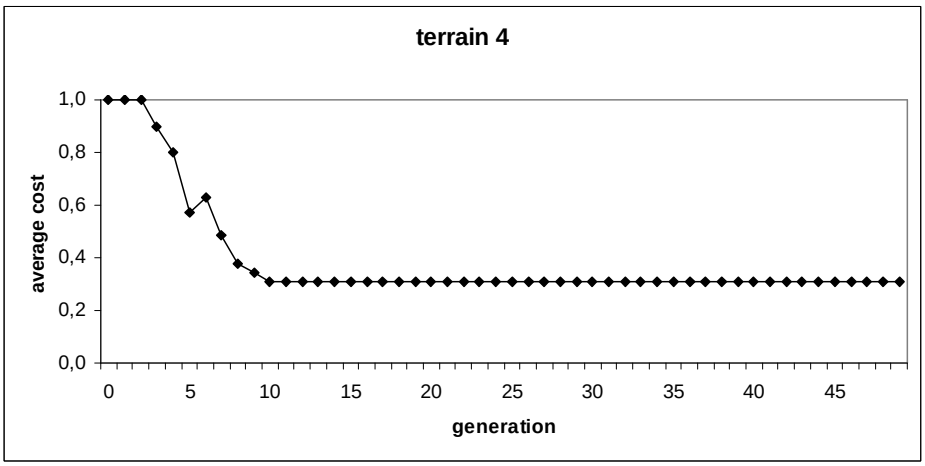




**Figure V.2.9 Average cost value of each generation for four terrains**

### **V.2.3.** Experiments for Setting HH Parameters

For hyper-heuristics, two factors affecting the performance of the method are tested and the performance analysis of each different method is evaluated. The first factor is the heuristic selection method which decides how to determine which heuristic is applied to the individual during hyper-heuristic phase. The methods considered in our study include *simple random*, *random descent*, *random*

*permutation descent*, *greedy approach* and three choice function implementations including *straight choice*, *ranked choice* and *roulette choice*.

The other factor is the acceptance criteria of the generated solutions. After applying the low-level heuristic selected by heuristic selection method to the solution, the acceptance criteria decides whether the solution is accepted or not. These methods include *Only Improving* (OI) which accepts the solutions if it improves the current solution, *All Moves* (AM) which accepts all new solutions and *Improving or Equal* (IE) which accepts the solutions improving the current solution or having same fitness. Moreover, the number of control points in each path is also tested for different values.

| **Parameter** | **Methods to be tested** | | | | | | |
|---|---|---|---|---|---|---|---|
| Heuristic selection | Simple random | Random descent | Random permutation descent | Greedy approach | Straight choice | Ranked choice | Roulette choice |
| Acceptance criteria | OI | AM | | IE | | | |
| # of control points | 5 | 7 | | 10 | 15 | | |

**Table V.2-5 Hyper-Heuristic Parameter Setting**

HH tests have also been executed for four terrain structures. The different number of control points gives similar results as in GA and PSO algorithms; therefore we used 5 control points for Terrain 1 and Terrain 3 and 7 control points for Terrain 2 and Terrain 4. The normalized average cost values at the end of the generation 1000 are given in Table V.2-6.

| | Terrain 1 | Terrain 2 | Terrain 3 | Terrain 4 |
|---|---|---|---|---|
| Simple Random-OI | 0.050866 | 0.113197 | 0.051072 | 0.066495 |
| Simple Random-AM | 0.357393 | 0.550951 | 0.341367 | 0.403325 |
| Simple Random-IE | 0.047180 | 0.168060 | 0.113349 | 0.113527 |
| Random Descent-OI | 0.045310 | 0.132330 | 0.220417 | 0.075613 |
| Random Descent-AM | 0.345818 | 0.546300 | 0.379887 | 0.437596 |
| Random Descent-IE | 0.077422 | 0.160925 | 0.082062 | 0.061243 |
| Random Permutation Descent-OI | 0.056459 | 0.094894 | 0.124165 | 0.061691 |
| Random Permutation Descent-AM | 0.389912 | 0.537905 | 0.347339 | 0.470409 |
| Random Permutation Descent-IE | 0.072452 | 0.139177 | 0.108926 | 0.116994 |
| Greedy Approach-OI | 0.190661 | 0.393703 | 0.373611 | 0.332378 |
| Greedy Approach-AM | 0.244594 | 0.297539 | 0.362993 | 0.245191 |
| Greedy Approach-IE | 0.364430 | 0.353737 | 0.342888 | 0.262177 |
| Straight Choice-OI | 0.077666 | 0.227836 | 0.190976 | 0.092692 |
| Straight Choice-AM | 0.188607 | 0.220226 | 0.190458 | 0.222814 |
| Straight Choice-IE | **0.039236** | 0.188039 | 0.243409 | 0.131056 |
| Ranked Choice-OI | 0.090062 | 0.154903 | 0.099861 | 0.203870 |
| Ranked Choice-AM | 0.388654 | 0.310685 | 0.460876 | 0.389910 |
| Ranked Choice-IE | 0.063940 | **0.094581** | **0.049109** | **0.058953** |
| Roulette Choice-OI | 0.115617 | 0.213261 | 0.220292 | 0.190607 |
| Roulette Choice-AM | 0.236298 | 0.338874 | 0.318568 | 0.167330 |
| Roulette Choice-IE | 0.098018 | 0.198870 | 0.186435 | 0.228415 |

**Table V.2-6 Average cost values obtained by the Hyper-Heuristic methods**

As seen in Table V.2-6, choice function based approaches which accept improving or equal solutions (IE) are better than the other approaches for all terrain structures. In general, AM acceptance method gives the worst results with the maximum cost values. Random approaches do not have the same performance as choice function based approaches. Based on the pre-experimentation phase, hyper-heuristic settings are given in Table V.2-7.

| **Parameter** | **Setting** |
|---|---|
| Heuristic selection | Straight choice or<br>Ranked choice |
| Acceptance criteria | Improving or Equal (IE) |
| # of control points | 5 for easy terrains<br>7 for difficult terrains |

**Table V.2-7 Final settings for the Hyper-Heuristics**

## V.3. COMPARISON STUDY

By determining the parameters of three proposed methods for UAV path planner, the experiments have been established for the comparison of the algorithms. For this part of the experimental study, additional terrains created by midpoint and hill algorithms are added to the experimental study (see Figure V.3.1). Each algorithm is executed for 30 times for each terrain structure.



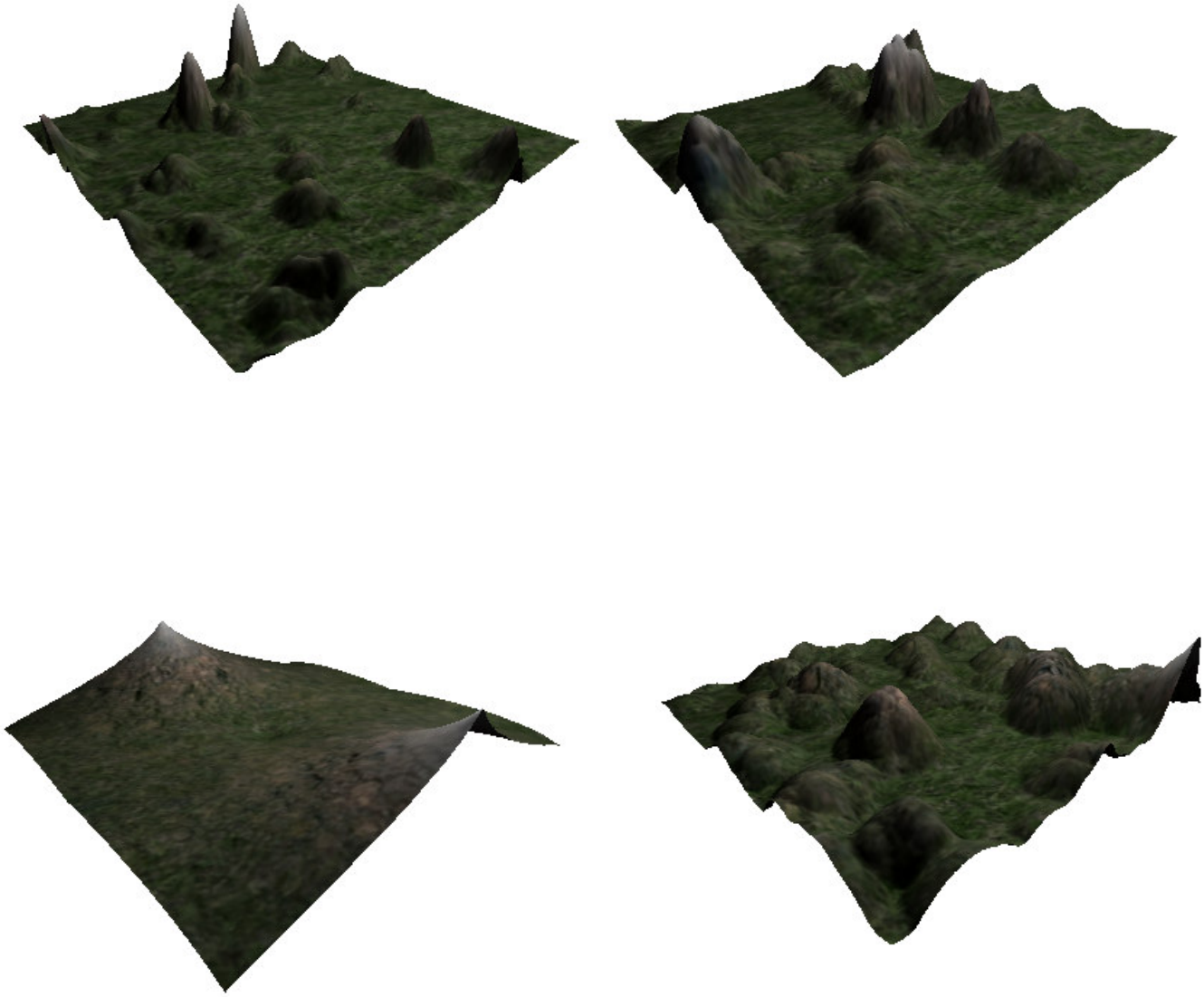


**Figure V.3.1 New Terrains for Comparison Study (numbered as Terrain5, Terrain6, Terrain7, Terrain8, Terrain9, Terrain10 respectively)**

The average normalized cost values of 30 different runs of each algorithm for the terrains are given in Table V.3-2. It can be figured out that the best results are achieved by Genetic Algorithms and Hyper-heuristics. While the average cost values of GA are the lowest ones for most of the terrain structures used in our experimental study, Hyper-heuristics gives better results for the most of the harder terrains including terrain 4, terrain 5 and terrain 10. To deal with harder terrain structures, HH is generally better than the other metaheuristics due to its set of low-level heuristics with different characteristics. The standard deviations of the cost values obtained by 30 different runs of three algorithms are also given in Table V.3-3. In

general, PSO show much worse performance than GA and HH. It's also observed that PSO is more successful for easier terrains (such as terrain 1 and terrain 6) than HH, since the algorithm can easily find the solution by approaching the random generated feasible best result of the population and not go away from the solution found during its execution.

| ***Terrain*** | ***Difficulty Level*** |
|---|---|
| Terrain1 | Easy |
| Terrain2 | Hard |
| Terrain3 | Easy |
| Terrain4 | Hard |
| Terrain5 | Hard |
| Terrain6 | Easy |
| Terrain7 | Easy |
| Terrain8 | Hard |
| Terrain9 | Easy |
| Terrain10 | Hard |

**Table V.3-1 Difficulty Levels of Terrain Used in the Experiments**

| | **GA** | **PSO** | **HH** |
|---|---|---|---|
| Terrain 1 | **0.07388747** | 0.11372085 | 0.11644246 |
| Terrain 2 | **0.06406619** | 0.10511967 | 0.14800588 |
| Terrain 3 | **0.07989536** | 0.12861760 | 0.20056592 |
| Terrain 4 | 0.11393455 | 0.19388939 | **0.07490808** |
| Terrain 5 | 0.12092739 | 0.12136236 | **0.11748002** |
| Terrain 6 | **0.06415956** | 0.11716578 | 0.20296123 |
| Terrain 7 | **0.02902435** | 0.04790236 | 0.13001969 |
| Terrain 8 | **0.08514292** | 0.15091295 | 0.20185281 |
| Terrain 9 | **0.09512903** | 0.11991147 | 0.10855423 |
| Terrain 10 | 0.12724021 | 0.12823793 | **0.05565815** |

**Table V.3-2 Average Cost Values of Algorithms for Different Terrains**

| | GA | PSO | HH |
|---|---|---|---|
| Terrain 1 | 0.04528386 | 0.05036542 | 0.11158013 |
| Terrain 2 | 0.03852906 | 0.02319679 | 0.09494684 |
| Terrain 3 | 0.04999550 | 0.09268936 | 0.13459508 |
| Terrain 4 | 0.04560835 | 0.11042816 | 0.09830751 |
| Terrain 5 | 0.05251748 | 0.08214498 | 0.10833708 |
| Terrain 6 | 0.03784570 | 0.02492691 | 0.13615052 |
| Terrain 7 | 0.04647378 | 0.08338414 | 0.14567411 |
| Terrain 8 | 0.05357843 | 0.08261034 | 0.10285532 |
| Terrain 9 | 0.03256975 | 0.05781413 | 0.08944678 |
| Terrain 10 | 0.04058415 | 0.08712944 | 0.06888966 |

**Table V.3-3 The Standard Deviation of Cost Values of Algorithms**

The means cost values (of 30 different runs) and their standard distributions are shown in Figure V.3.2 and Figure V.3.3, respectively. These figures are obtained by calculating the mean of the runs by three different algorithms including GA, PSO and HH. The mean values ( $\overline{X}$ ) are plotted by dots, the vertical lines indicate the distribution of these runs with $\overline{X} \pm s$ where $s$ is the standard deviation of the results. Figure V.3.2 shows the means and distributions for an easy terrain and Figure V.3.3 shows the means and distributions for an hard terrain. For the other terrains similar results are obtained. In these figures, it can be seen that GA performs better than the other two algorithms for both cases.

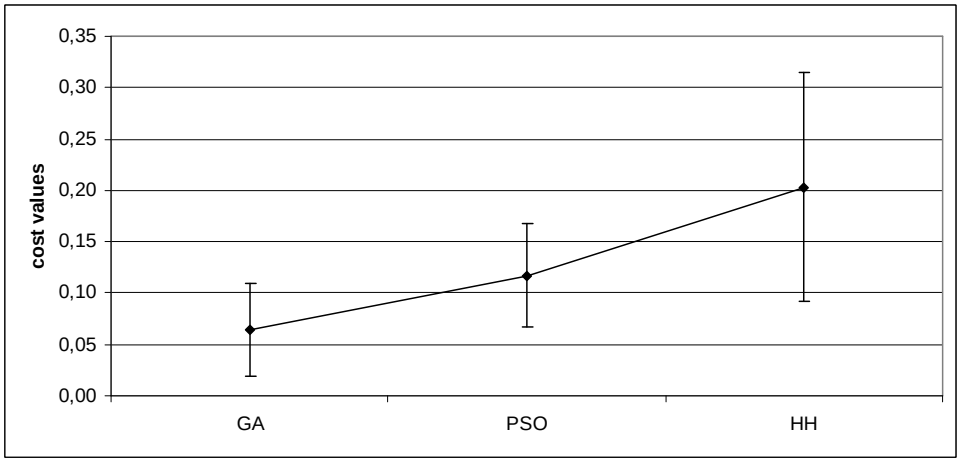



**Figure V.3.2 The Means and Distributions of the Normalized Cost Values for the Algorithms for an Easy Terrain (Terrain6)**

In Figure V.3.2, it can be seen also that PSO shows better performance than HH while in Figure V.3.3 HH has lower cost values which means better performance. For harder terrains, PSO gives much worse results than HH and GA.

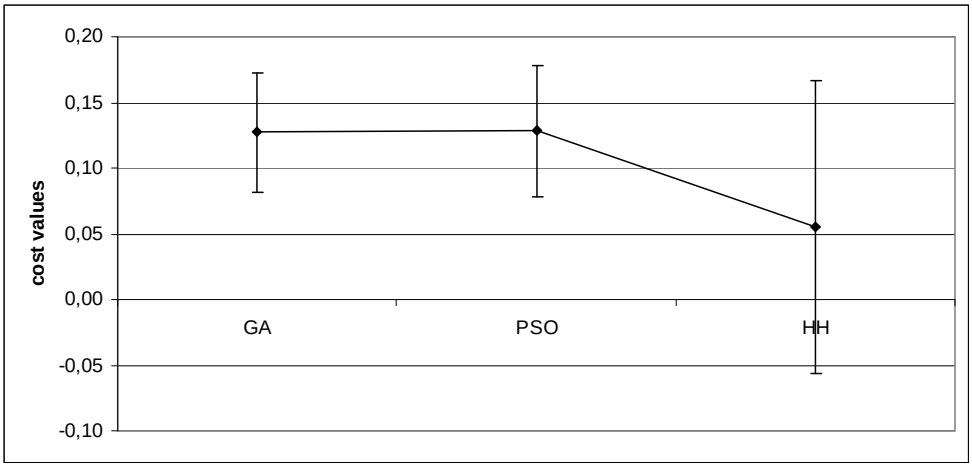



**Figure V.3.3 The Means and Distributions of the Normalized Cost Values for the Algorithms for an Hard Terrains (Terrain10)**

The average running times for 30 terrain structures obtained by three algorithms are given in Table V.3-4. The PSO algorithm requires the minimum time among them in most of cases, while more time is necessary for the HH method. Since HH method uses choice function based approaches, it takes long time to calculate the choice function values of each mutation type. PSO algorithm takes less time than GA because of its simplicity which does not have large computations.

| | **GA** | **PSO** | **HH** |
|---|---|---|---|
| Terrain 1 | 5015.9 | **3780.3** | 4459.3 |
| Terrain 2 | 2168.2 | **1087.3** | 1918.3 |
| Terrain 3 | 2821.1 | **1869.4** | 3281.3 |
| Terrain 4 | **4808.1** | 5238.1 | 4967.6 |
| Terrain 5 | **3888.4** | 3987.0 | 3944.1 |
| Terrain 6 | 2398.8 | **1052.4** | 1828.1 |
| Terrain 7 | 2147.6 | **1815.6** | 5138.6 |
| Terrain 8 | 2010.5 | **1191.6** | 2542.2 |
| Terrain 9 | 2959.0 | **2190.7** | 3279.6 |
| Terrain 10 | **5366.4** | 6195.7 | 5690.1 |

**Table V.3-4 The Average Running Times (in msecs) of Algorithms for Different Terrains**

In Figure V.3.4, sample paths generated by our path planner framework is shown. The path above is generated for Terrain1 and has five control points, while the other path is generated for Terrain4 which has 7 control points including source and destination points to deal with hard terrain structure.

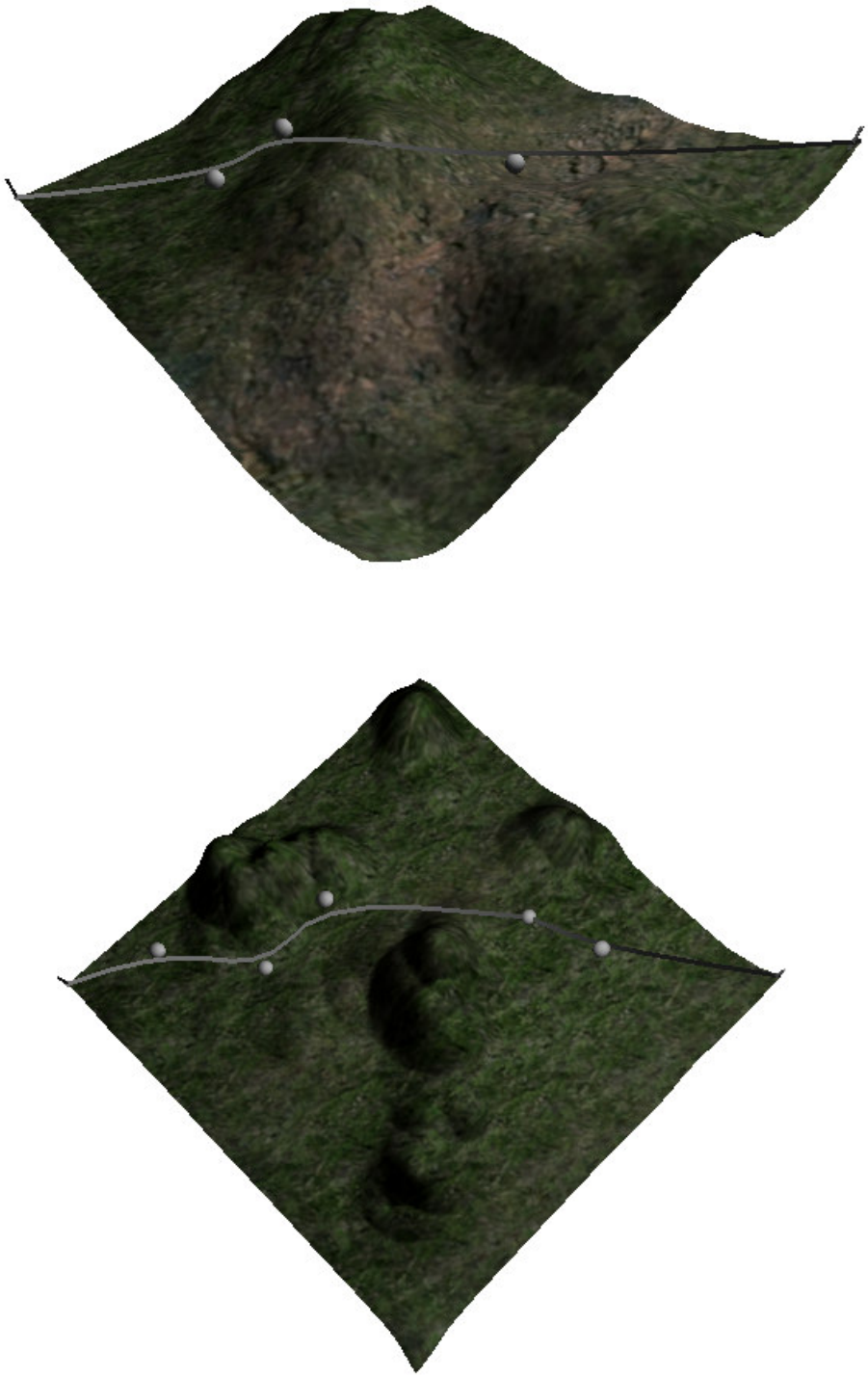


**Figure V.3.4 Sample Paths for Sample Terrains**

# PART VI

## CONCLUSION AND FUTURE WORK

In this study, we implemented different metaheuristic techniques including Genetic Algorithms, Particle Swarm Optimization and Hyper-Heuristics to develop a path planner for Unmanned Aerial Vehicles by considering different objectives and constraints. Specifically, generating feasible paths without terrain collision, generating paths within the desired minimum and maximum distance to terrain, generating paths that provide UAV to maneuver with an angle greater than allowable minimum curvature radius, and generating paths that avoid threat zone in a terrain are the set of constraints in our study. Minimizing the length of the path from source to destination and generating a flight envelope in a given line of sight are the objectives in our study.

In our framework, UAV path planning problem are solved with three techniques, which are Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Hyper-Heuristics (HH). The proposed techniques are compared with different terrains which are generated by different terrain generation algorithms. The performance of each algorithm is presented in detail and the comparison of the algorithms is performed by using statistical data gained from the experimentation phase.

The experimental study shows that the Genetic Algorithms generally have performed better than both PSO and HH methods with reasonable running times. The proposed mutation and crossover types provide better performance (lower total cost values) for different terrain structures. However, HH methods gives better results than other methods for harder terrain structures. Choice function based HH methods perform well to deal with difficult terrain structures by its low-level heuristics.

As a future work, PSO Algorithm and HH can be improved with more sophisticated methods. Additionally, the proposed techniques are planned to apply for online UAV path planning problem.

# REFERENCES


**[1]** http://en.wikipedia.org/wiki/Unmanned_aerial_vehicle, **(2008)**.

**[2]** Gerald Farin: "*Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*", Academic Press, Fifth Edition, Morgan Kaufmann, **2001**.

**[3]** A.E. Eiben; J.E. Smith: "*Introduction to Evolutionary Computing*", Springer (Natural Computing Series), **2003**.

**[4]** http://www.swarmintelligence.org/, **(2008)**.

**[5]** Edmund Burke; Graham Kendall; Jim Newall; Emma Hart; Peter Ross; Sonia Schulenburg: "Hyper-Heuristics: An Emerging Direction in Modern Search Technology", Handbook of Metaheuristics, Kluwer Academic Publishers, **(2003)**, 457-474

**[6]** Peter Cowling; Graham Kendall; Eric Soubeiga: "A Hyperheuristic Approach to Scheduling a Sales Summit", International Conference on Practice and Theory of Automated Timetabling, **2000**, 176-190

**[7]** Brian J. Capozzi: "Evolution-Based Path Planning and Management for Autonomous Vehicles", *Doctor of Philosophy Thesis*, University of Washington **(2001)** 112-151.

**[8]** Anargyros N. Kostaras, Ioannis K. Nikolos, Nikos C. Tsourveloudis, and Kimon P. Valavanis: "Evolutionary Algorithm Based Online Path Planner For UAV Navigation", Proceedings of the 10th Mediterranean Conference on Control and Automation - MED2002, July 9-12, 2002.

**[9]** UAV Annual Report FY 1996, **(1996)** http://fas.org/irp/agency/daro/uav96/content.html

**[10]** US DoD Unmanned Systems Roadmap:2007-2032, **(2007)**

**[11]** Yao-hong Qu; Quan Pan; Jian-guo Yan: "Flight Path Planning of UAV Based on Heuristically Search and Genetic Algorithms", IEEE **(2005)** 45-46.

**[12]** Jeremi Gancet; Gautier Hattenberger; Rachid Alami; Simon Lacroix: "Task planning and control for a multi-UAV system", *IEEE/RSJ International Conference*, on Volume , Issue , 2-6 Aug. **2005**, 1017 – 1022.

**[13]** Bruno Sinopoli; Mario Micheli; Gianluca Donatot; T.John Koo: "Vision Based Navigation for an Unmanned Aerial Vehicle", *Proceedings of the 2001 IEEE International Conference on Robotics & Automation*, May 21-26, **2001**

**[14]** Atilla Doğan: "Probabilistic Path Planning For UAVs", *2nd AIAA Unmanned Unlimited Systems, Technologies, and Operations — Aerospace,* 15 - 18 September **2003**

**[15]** Barbara Pfeiffer; Rajan Batta; Kathrin Klamroth; Rakesh Nagi: "Path Planning for UAVs in the Presence of Threat Zones Using Probabilistic Modeling", *IEEE Transactions on Automatic Control*, No 43 **(2007)**, 278-283.

**[16]** Tom Schouwenaars; Jonathan How; Eric Feron: "Receding Horizon Path Planning with Implicit Safety Guarantees", *Proceeding of the 2004 American Control Conference*, June 30 - July 2, **2004**

**[17]** Ellis King; Yoshiaki Kuwata; Jonathan P. How: "Experimental Demonstration of Coordinated Control for Multi-Vehicle Teams", *International Journal of Systems Science*, Vol. 37, No. 6, 15 May **2006**, 385–398

**[18]** Mehdi Alighanbari; Jonathan P. How: "Cooperative Task Assignment of Unmanned Aerial Vehicles in Adversarial Environments", *American Control Conference Proceedings*, vol. 7 , Issue , 8-10 June **2005** 4661 – 4666

**[19]** Suman Chakravorty;John L. Junkins: "A Methodology for Intelligent Path Planning", *Proceedings of the 2005 IEEE International Symposium on Intelligent Control*, June 27-29, **2005**, 592-597

**[20]** http://en.wikipedia.org/wiki/Markov_decision_process, **(2008)**

**[21]** Jack Langelaan; Steve Rock: "Navigation of Small UAVs Operating in Forests", IAA *Guidance, Navigation and Control Conference*, **2004**

**[22]** Dah-Jye Lee; Randal W. Beard; Paul C. Merrell; Pengcheng Zhan: "See and avoidance behaviors for autonomous navigation", *SPIE Optics East, Robotics Technologies and Architectures, Mobile Robot XVII*, vol. 5609-05, October 25-28, **2004**.

**[23]** http://en.wikipedia.org/wiki/Differential_evolution, **(2008)**

**[24]** Ioannis K. Nikolos; Athina N. Brintaki: "Coordinated UAV Path Planning Using Differential Evolution", *Proceedings of the 13thMediterranean Conference on Control and Automation*, June 27-29, **2005**

**[25]** Jang-Ho Lee; Byoung-Mun Min; Eung-Tai Kim: "Autopilot Design of Tilt-rotor UAV Using Particle Swarm Optimization Method", *International Conference on Control, Automation and Systems* **2007**

**[26]** Jung Leng Foo; Jared S. Knutzon; James H. Oliver; Eliot H. Winer: "Three-Dimensional Path Planning of Unmanned Aerial Vehicles Using Particle Swarm Optimization", *AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 6 – 8 September, **2006**.

**[27]** Shibo Li; Xiuxia Sun; Yuejian Xu: "Particle Swarm Optimization for Route Planning of Unmanned Aerial Vehicles", *Proceedings of IEEE International Conference on Information Acquisition*, August 20 – 23, **2006**.

**[28]** David Rathbun, Ph.D.;Brian Capozzi, Ph.D.: "Evolutionary Approaches to Path Planning Through Uncertain Environments", *Proceedings of the AIAA Unmanned Unlimited Conference*, May, **2002**.

**[29]** Juan Carlos Rubio; Sean Kragelund: "The Trans-Pacific Crossing:Long Range Adaptive Path Planning for UAVs Through Variable Wind Fields", *Digital Avionics Systems Conference*, The 22nd Volume 2, Issue , 12-16 Oct. **2003** 8.B.4 - 81-12 vol.2

**[30]** David Rathbun; Sean Kragelund; Anawat Pongpunwattana;Brian Capozzi: "An Evolution Based Path Planning Algorithm for Autonomous Motion of a UAV Through Uncertain Environments", *Digital Avionics Systems Conference Proceedings*, **2002** Volume: 2

**[31]** Juan Carlos Rubio Torroella: "Long Range Evolution-based Path Planning for UAVs through Realistic Weather Environments", *Master of Science Thesis in Aeronautics and Astronautics, University of Washington*, **2004**, 22-26.

**[32]** Ioannis K. Nikolos; Nikos C. Tsourveloudis; and Kimon P. Valavanis: "Evolutionary Algorithm Based 3-D Path Planner for UAV Navigation", *Proceedings of the 9th Mediterranean Conf. on Control and Automation*, **2001**.

**[33]** Ioannis K. Nikolos; Kimon P. Valavanis; Nikos C. Tsourveloudis; Anargyros N. Kostaras: "Evolutionary Algorithm Based Offline/Online Path Planner for UAV Navigation", IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics, vol.33, no. 6, pp. 898-912, **2003**.

**[34]** J.H. Holland; "Adaptation in Natural and Artificial Systems", *University of Michigan Press, Ann Arbor, Michigan*; **(1975)** re-issued by MIT Press (1992)

**[35]** James Kennedy;Russell Eberhart: "Particle Swarm Optimization", *Proceedings of the IEEE International. Conference on Neural Networks*, **1995**, 1942 – 1948.

**[36]** Marco Dorigo: "Swarm Intelligence", *Springer New York*, **(2007)**

**[37]** J. Cruz; G. Chen; D. Li; X. Wang: "Particle Swarm Optimization for Resource Allocation in UAV Cooperative Control", *AIAA Guidance, Navigation, and Control Conference and Exhibit*, **2004**

**[38]** Yuhui Shi; Russell C. Eberhart: "Parameter Selection in Particle Swarm Optimization", *Proceedings of the 7th International Conference on Evolutionary Programming VII* , **(1998)**, 591-600

**[39]** Edmund Burke; Graham Kendall; Jim Newall; Emma Hart; Peter Ross; Sonia Schulenburg: "Hyper-Heuristics: An Emerging Direction in Modern Search Technology", Handbook of Metaheuristics, Kluwer Academic Publishers, **(2003)**, 457-474

**[40]** David H. Wolpert; William G. Macready: "No Free Lunch Theorems for Optimization", IEEE Transactions on Evolutionary Computation, **(1996)**

**[41]** Eric Soubeiga: "Development and Application of Hyperheuristics to Personnel Scheduling", PhD Thesis in School of Computer Science and Information Technology The University of Nottingham, **(2003)**, 4

**[42]** Eric Soubeiga: "Development and Application of Hyperheuristics to Personnel Scheduling", PhD Thesis in School of Computer Science and Information Technology The University of Nottingham, **(2003)**, 87

**[43]** Dr Gabriela Ochoa: "Improvement or Local-Search Hyper-heuristics", Istanbul Technical University Summer School Lecture Notes, 31st Jul – 3rd Aug **2007**

**[44]** Peter Cowling; Graham Kendall; Eric Soubeiga: "A Hyperheuristic Approach to Scheduling a Sales Summit", International Conference on Practice and Theory of Automated Timetabling, **2000**, 176-190

**[45]** Alexander Nareyek: "Choosing Search Heuristics by a Non-Stationary Reinforcement Learning", *Metaheuristics: Computer Decision Making*, Kluwer Academic **(2001)** 523-544

**[46]** Eric Soubeiga: "Development and Application of Hyperheuristics to Personnel Scheduling", PhD Thesis in School of Computer Science and Information Technology The University of Nottingham, **(2003)**

**[47]** Sugih Jamin: "Terrain Generation", University of Michigan, Computer Game Design and Development Lecture Notes, **(2006)**

**[48]** http://www.robot-frog.com/3d/hills/heightmaps.html, **(2008)**

**[49]** H. Burchardt; R. Salomon: "Implementation of Path Planning using Genetic Algorithms on Mobile Robots", *Congress on Evolutionary Computation, CEC*, **2006**, 1831-1836.

**[50]** D. Rathbun; S. Kragelund; A. Pongpunwattana; B. Capozzi: "An Evolution Based Path Planning Algorithm For Autonomous Motion of a UAV Through Uncertain Environments", *Proceedings of Digital Avionics Systems Conference*, vol. 2, 8D2-1-8D2-12, **2002**.

**[51]** http://mathworld.wolfram.com/B-Spline.html, **(2008)**

**[52]** Eric Soubeiga: "Development and Application of Hyperheuristics to Personnel Scheduling", PhD Thesis in School of Computer Science and Information Technology The University of Nottingham, **(2003)**, 92

**[53]** jME graphics API: http://www.jmonkeyengine.com/, **(2008)**

**[54]** Matthias Grumet: "Terrain Modeling", Report, Institute of Computer Graphics and Algorithms Vienna University of Technology **(2004)**

**[55]** Terrain Generation Tutorial: Hill Algorithm, http://www.robot-frog.com/3d/hills/hill.html, **(2008)**

**[56]** Jason Shankel: "Fractal Terrain Generation - Midpoint Displacement", *Game Programming Gems*, Charles River Media, **(2000)**, 503-507.

**[57]** http://www.gameprogrammer.com/fractal.html, **2008**

**[58]** Jason Shankel: "Fractal Terrain Generation - Fault Formation", *Game Programming Gems*, Charles River Media, **(2000)**, 499-502

**[59]** Jason Shankel (Maxis): "Fractal Terrain Generation - Particle Deposition", *Game Programming Gems*, Charles River Media, **(2000)**, 508-511

**[60]** Fernando Rinaldi: "Basics of Molecular Beam Epitaxy (MBE)", Annual Report, Ulm University Institute of Optoelectronics **(2002)**

# RESUME


## I Ş I L  H A S I R C I O Ğ L U

İcadiye Mah. Çamlıca Cad. Şile Apt. No:5/8  Üsküdar/İSTANBUL
**Telefon (Ev): 0 216 532 62 97/ Telefon (Cep) : 0 532 307 48 36**
**e-mail:** isilcse@hotmail.com


### EDUCATION

| | | |
|---|---|---|
| **2005 -** | **Marmara University Computer Engineering** | |
| MS | | |
| **1999 - 2004** | **Marmara University Computer Engineering** | |
| BS | | |
| **1995 - 1999** | **Kadıköy Habire Yahşi High School** | |
| High School | | |

### WORK EXPERIENCE

- **August 2004- Present: TÜBİTAK UEKAE**, Researcher
- **April 2004-June 2004 : TSOFT Bilişim Teknolojileri**, part time software developer
- **October 2003 - January 2004 : İkiNokta Bilişim A.Ş.**, part time software developer
- **August 2003 - September 2003 :Arçelik A.Ş.- Bilişim Sistemleri Direktörlüğü**, Summer Intern
- **July 2002-August 2002: T-Systems Bilişim Teknolojileri A.Ş.**, Summer Intern

### PERSONAL INFORMATION

Birth Date and Place : 28/11/1981 - Acıpayam