Genetic Programming to Optimise 3D Trajectories

André Kotze

Thesis submitted in partial fulfilment of the requirements for the Degree of Master of Science in Geospatial Technologies

Supervised by:

Dr. Carlos Granell Canut
Institute of New Imaging Technologies
Universitat Jaume I
Castellón de la Plana, Spain

Co-supervised by:

Dr. Vítor Duarte dos Santos
NOVA Information Management School
Universidade Nova de Lisboa
Lisboa, Portugal

Moritz Hildemann
Institute for Geoinformatics
Westfälische Wilhelms-Universität
Münster, Germany

February 2023

Declaration of Academic Integrity

I hereby confirm that this thesis "Genetic Programming to Optimise 3D Trajectories" is solely my own work and that I have used no sources or aids other than the ones stated. All passages in my thesis for which other sources, including electronic media, have been used, be it direct quotes or content references, have been acknowledged as such and the sources cited. I agree to have my thesis checked in order to rule out potential similarities with other works and to have my thesis stored in a database for this purpose.

André Kotze
Castellón de la Plana
17 February 2023

Acknowledgements

I would like to express my deepest gratitude to the following people for enabling the completion of this project:

- Prof. Dr. Marco Painho at NOVA University Lisbon, Dr. Christoph Brox at the University of Münster, and Prof. Dr. Joaquín Huerta at Jaume I University, for their work in coordinating the joint master's program. It has been one of the most significant experiences of my academic, social, and professional life.
- My supervisors Dr. Carlos Granell, Dr. Vítor Santos and Moritz Hildemann for their time, support and guidance throughout the duration of this work. Without your patient advice this thesis would not be nearly as complete or coherent. An additional thanks to Moritz for the thesis topic, the ideas, and the data.
- My professors, who have inspired me to keep seeking, exploring, thinking and learning. Thank you for your passion and dedication to teaching.
- My parents for their love and support. Every opportunity I've had I owe to you.
- My sister for believing in me, encouraging me, and always noting *Good, but can do better*, and without whom I would never have applied to this program. Thank you for the honesty, the proofreading, and the formatting.
- My friends, old and new. This adventure would not be the same without you. Thanks for your companionship and advice when I most needed it.
- The open source community, who have allowed me to see further by standing on their shoulders, who have freely shared their knowledge, expertise, and code with the world. Without your tireless efforts this work would not have been possible. Your generosity and commitment to the ideals of open access, collaboration, and innovation are an inspiration.

Abstract

Trajectory optimisation is a method of finding the optimal route connecting a start and end point. The suitability of a trajectory depends on non-intersection with any obstacles as well as predefined performance metrics. In the context of UAVs, the goal is to minimise the cost of the route, in terms of energy or time, while avoiding restricted flight zones. Artificial intelligence techniques including evolutionary computation have been applied to trajectory optimisation with various degrees of success. This thesis explores the use of genetic programming (GP) to optimise trajectories in 3D space, by encoding 3D geographic trajectories as syntax trees representing a curve. A comprehensive review of the relevant literature is presented, covering the theory and techniques of GP, as well as the principles and challenges of 3D trajectory optimisation. The main contribution of this work is the development and implementation of a novel GP algorithm using function trees to encode 3D geographical trajectories. The trajectories are validated and evaluated using a real-world dataset and multiple objectives. The results demonstrate the effectiveness of the proposed algorithm, which outperforms existing methods in terms of speed, automaticity, and robustness. Finally, insights and recommendations for future research in this area are provided, highlighting the potential for GP to be applied to other complex optimisation problems in engineering and science.

Keywords: genetic programming, evolutionary algorithms, trajectory optimisation, route planning

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background and Problem Definition

Pathfinding (also referred to as pathing or routing) in 2-dimensional (2D) space is widely applied to comparatively simple navigational and logistical problems that may or may not be confined to segments that are part of an existing network of pathways (Garip et al., 2022). The problems become more complex with the introduction of additional considerations and constraints, such as multiple points (Travelling Salesman Problem)(Pezer, 2016), multiple agents (Vehicle Routing Problem) (Baker & Ayechew, 2003) or dynamism (transient obstacles, alternative routes, Dynamic Vehicle Routing Problem) (Kim et al., 2017).

Path planning in 3-dimensional (3D) space is an ongoing research topic that expands upon simpler 2D routing. This more complex problem has applications in path planning for autonomous or remotely-operated vehicles, such as Unmanned Aerial Vehicles (UAVs) (Meng & Xin, 2010; Wang et al., 2015) or Unmanned Underwater Vehicles (UUVs) (An, 2018; Balicki, 2006). In the case of unmanned aircraft, height (relative or absolute) is incorporated as the third spatial dimension. For unmanned submarine vehicles, the relevant dimension is depth.

Optimising paths in a space where transport infrastructure is irrelevant (aerial, submarine, subterranean, extraterrestrial) is complicated by the fact that the number of possible routes is infinite (Hu et al., 2004). In addition to minimising the path cost in terms of time or distance, the path also needs to avoid physical obstacles and restricted areas. These barrier zones can have legal, cost- or safety-related significance. For an exact optimisation that sequentially evaluates every possible route it is impossible to find the optimum in finite time due to the immense computational requirements for problems with "large search spaces with large numbers of potential solutions" (Behzadi & Alesheikh, 2008). It is therefore needed to optimise the search and sampling method within the solution space.

Heuristic optimisation refers to a solution of maximum efficiency according to specified criteria, which cannot be guaranteed to be the optimal solution. In the case that a solution x is found within time t, and significant additional solving time does not yield significantly better solutions, solution x is heuristically optimal. This approach is more practical than exact optimisation, which continues until the single, technically most perfect solution is obtained *i.e.* an optimal solution is guaranteed. In real-world applications such as free-flight routing (Hu et al., 2004) time is of the essence. In these situations a quicker, sub-optimal solution is more valuable than a perfectly optimal one that cannot be found in finite time.

In the realm of heuristic optimisation, evolutionary computation is a biologically inspired inexact approach to optimisation. Algorithms mimic natural evolution by testing the aptness of individuals within a population over a number of generations, and applying selective pressures via a cost (or fitness) function. The procedure to find a heuristic solution is referred to as a metaheuristic. Therefore, nature-inspired metaheuristic methods such as evolutionary algorithms and genetic algorithms, are designed to find a sufficiently good solution by subsampling from the enormous number of possible solutions in 3D space, instead of exploring them sequentially.

A research gap exists in the representation of heuristic geometries in a chromosome. Previous studies (Hildemann & Verstegen, 2023) using Genetic Algorithms (GA) have used a directly translated chromosome, with individual genes consisting of the point coordinates that ul-

timately comprise the linestring. These solutions, while very effective, have an enormous computational demand. Path outputs from the optimisation method need to be validated to ensure that no barriers are intersected, and since validation is required for every possible solution the number of validations can be a significant and limiting factor due to its effect on computation time. Using the same evolutionary algorithm, it is possible to change the method of chromosome encoding to a more efficient format.

The aim of this thesis is to explore the trajectory optimisation problem in 3-dimensional space using Genetic Programming (GP), an evolutionary computation technique that has been applied to similar problems in two dimensions (Cakir, 2015; Hanshar & Ombuki-Berman, 2007) and three dimensions (Meng & Xin, 2010; Wang et al., 2015; Yang et al., 2016). In the GP approach, solutions are represented by functions that can be evaluated *i.e.* programs. These programs are encoded into chromosomes, which can be modified by changing the constituent genes. Solutions are evaluated according to their fitness in terms of cost and efficiency using a cost function. Then, a subset of the fittest solutions is selected to produce the next generation. As chromosomes reproduce, genes are exchanged, combined and mutated in a process analogous to biological evolution (Behzadi & Alesheikh, 2008), and successive generations are expected to yield better solutions, evolving towards an optimal trajectory.

One such option is encoding genes as terms within a function, with each path consisting of one function. This places the method in the domain of Genetic Programming (as opposed to GA) since the solutions consist of function trees, which are programmatic representations and can be manipulated like genetic programs. A segment of the 3D line represented by that function is then transformed into the problem space and evaluated. The validation criterion is non-intersection with barriers. The evaluation criterion is cost, in terms of distance or time, and depends on a number of factors including length and gradient. Trajectories therefore need to be converted into geographical linestrings to be validated and evaluated for their suitability in terms of travel cost and intersection with barriers, according to a cost function. Validation could occur within a GIS with routes and barriers modelled as features, or in another suitable framework.

With this focus on chromosome representation and efficient validation, the following research questions can be stated:

- Can functions encoding curves be effectively evolved to solve a minimisation problem?
- Can the speed of 3D geometric path optimisation be significantly increased by changing the method of path representation?
- How does the method of geometry processing affect processing speed?
- Can path length be quantified before translating the function tree into a geometry?

1.2 Objectives

The goal of this research is to design and evaluate a GP solution to the 3D trajectory problem, with the explicit objective of optimising routes in the presence of 3D obstacles and barriers.

To achieve this goal, the following intermediate objectives are defined:

- Conduct a thorough review of existing techniques and knowledge from the literature

- Select an appropriate data structure for 3D linestring and zone geometry storage, representation and computation
- Design a suitable chromosome representation for 3D linestrings that is small in size and robust against continuous modification
- Design a suitable cost function to evaluate and select solutions from every generation
- Calibrate GP parameters for the 3D routing problem

1.3 Importance and Relevance

Path planning is an integral part of robotic vehicle operation and has many current applications in operations research, distribution logistics, supply chain management and transportation (Behzadi & Alesheikh, 2008). Many modern exploratory and scientific expeditions, as well as commercial or private routes, are pre-planned and completely autonomously navigated by the drones involved. Although remotely-operated vehicles (ROVs) and remotely piloted aircraft (RPAs) are actively controlled by an expert, they can also benefit from optimal routes. These routes can assist the pilots by serving as a point of reference for maximum efficiency. Furthermore, UAVs are susceptible to electro-mechanical faults which can necessitate an emergency landing. In these situations generating a safe path within limited time is critical (Garg et al., 2015).

The main advantage of specifically 3-dimensional path planning is the ability to navigate more complex environments, with the possibility of utilising routes over or under no-go zones. An assumption in similar 2D problems is that danger zones extend to infinity in altitude or depth, and route optimisation outputs are bound to a cost surface. However, this is not the case in many scenarios (Hildemann & Verstegen, 2023) and with increasingly agile robotic vehicles the need to exploit such routes becomes clear. Examples in airspace include tunnels and arches, the undersides of bridges and other structures, the tops of buildings and trees, and any navigable zone beneath an overhang or "above" a no-go zone. Submarine examples include arches and caves. Establishing a limit on the extent of barriers allows many new routes to be considered in the optimisation process.

1.4 Thesis Outline

This section describes the research and development process, attempting to solve the path optimisation problem in 3D in the context of UAVs. The work can be broadly divided into 3 consecutive parts, commencing with an investigation into the specific details of the problem as well as exploring the various ways in which evolutionary computation has been applied to similar problems. Figure 1 shows a simple breakdown of the proposed workflow.

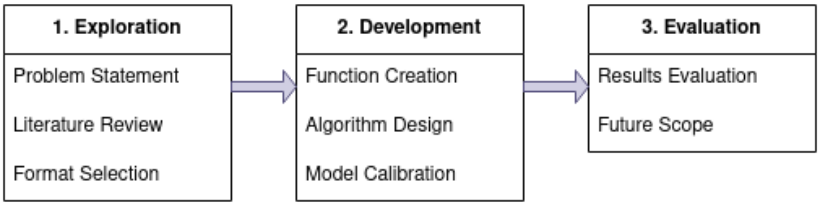


Figure 1: Study breakdown

The exploration phase starts with a comprehensive and thorough literature review, with the aim of finding relevant examples and case studies in the context of GP and geometric transformation (Section 2). It is also necessary to investigate already established differences in geometric processing of linestrings as stored sets of vertices versus functions.

The main considerations in the methodology design (Section 3) are scale and speed of processing. As chromosome size increases, processing time increases proportionally. The same is true for the size and scale of the barrier dataset, and the granularity of the solution path.

Frameworks need to be selected for the chromosome representation and the geographic representation of solutions. The speed of the validation process needs to be minimised, as the number of validations required may be significant. Several methods exist to validate outputs, and the most efficient routine will have to be identified through testing.

A demonstration of the results (Section 4) is followed by a discussion of their implications (Section 5). Finally, a summary of the research contribution and a selection of areas of possible future study is identified and discussed in the conclusion (Section 6).

2 Literature Review

A review of scientific publications related to the concepts and methods in this thesis has been conducted in order to construct a knowledge base and identify knowledge gaps. Focus is placed on peer-reviewed full papers that discuss pathfinding and route optimisation in general, as well as optimisation techniques and algorithms that have been or can be applied to 3-dimensional trajectories.

For trajectory optimisation specifically, an additional Systematic Literature Review (SLR) is conducted to ensure that all relevant research output and existing methods are known and understood before proceeding with the proposed methodology. The SLR also ensures that the conceptual and technical context of this research is optimally defined and illustrated.

2.1 Background Concepts

2.1.1 Pathfinding Algorithms

Pathfinding algorithms are search algorithms that aim to connect start and end points by a navigable path. The space in which this occurs can be either discrete or continuous. Discrete spaces (also known as graphs, subdivided spaces, grids or networks) consist of distinct nodes, samples or points connected by a finite set of edges, links or lines. Thus pathfinding is reducible to a topological problem with a finite set of possible configurations. Algorithms such as Dijkstra's use exhaustive means to find a solution by repeatedly searching adjacent nodes until the destination node is reached. The shortest path problem has the additional requirement of finding the optimal route between nodes *i.e.* minimising the total cost of the path. Other common problems in graphs are the travelling salesman problem (TSP) which aims to find the shortest route that visits every node once, and the Hamiltonian path problem, a special case of TSP that attempts to avoid ever using the same edge twice.

Artificial Neural Networks (ANN) have been applied to many optimisation problems, with overall success. They have an advantage over techniques such as Simplex or Lagrange in that they can solve non-linear problems. Some heuristics include ant colony optimisation (ACO) , simulated annealing and particle swarm optimisation (PSO) (Villarrubia et al., 2018). However, these methods are only applicable in discontinuous space. The trajectory optimisation problem under consideration is continuous, and an infinite set of configurations is possible. It is therefore necessary to use a search method that efficiently explores this unconstrained space in finite time *i.e.* without considering every possible configuration in the search space.

2.1.2 Evolutionary Algorithms

Also known as evolutionary computation (EC), this is a type of artificial intelligence that encompasses algorithms that mimic biological evolutionary processes. Global optimisation is achieved via variation and selection, in the same way that species adapt to environments over generations via natural selection. Evolutionary algorithms have historically been divided into four major paradigms: genetic algorithms (GA), genetic programming (GP), evolutionary programming (EP) and evolution strategy (ES). These methods differ in the manner in which solutions are represented as well as the selection methods and reproduction operators (Sivanandam & Deepa, 2008).

Evolutionary algorithms minimally require genetic operators that drive the diversification into the search space, and selection pressures that guide the adaptation towards a specific goal *i.e.* an adaptive system (Holland, 1975). These processes of variation and selection are what separate a parent population from its offspring in the next generation. Genetic operators seek to randomise traits by mixing the traits of two individuals (crossover) or mutating the traits within a single individual (mutation). This diversification is made to be beneficial, in accordance with the desired criteria, by the selection process. Individuals are evaluated by quantifying their fitness via a cost function, and are subsequently filtered in order to yield an offspring population that preserves the best traits of their parents. In this manner the solution is optimised, by repeated variation and selection through hundreds or thousands of generations.

Genetic algorithms have been used extensively for pathfinding and navigation problems in the context of optimisation (Alves et al., 2020; Kumar et al., 2014) and obstacle-avoidance (Ma et al., 2020; Rath et al., 2019). Hu et al. (2004) and Hildemann and Verstegen (2023) use chromosomes that assign a single gene to represent a single vertex of the linestring. In GA, solutions can be encoded as strings, numbers, binary sequences, matrices, symbols (Meng & Xin, 2010) or any other modifiable and encodable data structure.

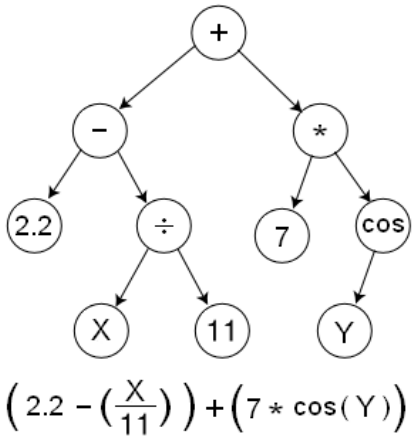


Figure 2: Example of a genetic program tree and the function it represents

Genetic programming is a systematic, domain-independent automatic problem solving method (Poli, Langdon, et al., 2008). It differs from GA in that a procedure, or program, is evolved in the optimising process. Instead of consisting of a directly encoded solution, the GP chromosome consists of an encoded function that evaluates to that solution. In linear genetic programming the evolving programs consist of a sequence of instructions processed in series. More commonly, tree-based GP is used (Koza, 1992), wherein programs are represented as function trees with nested subordinate trees (branches) spanning multiple levels (Fig. 2). During reproduction, linear genetic programs swap out or mutate single instructions, whereas whole nodes and branches are swapped or mutated in the case of function trees. This implies the potential for extremely high variance in the size of function trees as opposed to fixed-length linear programs, and thus the potential for greater diversity.

Several authors have tackled problems involving multiple agents or objectives by stacking

or nesting EC methods. Kala (2012) used co-evolutionary GP, a multi-level optimisation method, for path planning for multiple robots with the constraint of non-collision. It employs a master GA that evaluates the overall optimality while individual agents evolve their own GP solutions. Edison and Shima (2011) took on the cooperative multiple task assignment problem (CMTAP) in the context of multiple cooperating UAVs with kinematic constraints using GA.

A number of default parameter values have been empirically derived, or exist due to computational constraints. Koza (1992) suggests a tree height limit of 17 and a maximum number of generations of 50. Poli, Langdon, et al. (2008) pose some general recommendations, while emphasising that optimal parameters are highly dependent on the application. They suggest a run length of 10 to 50 generations claiming that the most productive search occurs this early, and like Koza (1992) they posit a population size of 500.

2.1.3 Trajectory Optimisation

Trajectory optimisation is a combination of pathfinding and minimisation. Because trajectories are not constrained to a pre-existing network, an additional degree of freedom exists in where paths can be placed (*cf: routing, pipe routing*) (Sandurkar & Chen, 1999). Thus, any graph-based solutions are inapplicable. Additional degrees of freedom can be introduced by considering additional dimensions, such as time or altitude. In this study, altitude (z-dimension) is considered and should facilitate the creation of paths over (or under) obstacles that are shorter than similar paths around the same obstacles. Theoretically, negative altitude or depth can be used similarly in the case of underwater navigation.

The optimal solution depends on arbitrarily chosen objectives, and can vary by application. Different goals in the same search space will yield different optimal solutions, and similarly different sets of goals will yield different Pareto fronts when considering multiple objectives. Objectives need to be quantifiable and some that have been considered in the case of unmanned vehicle trajectories include noise pollution, scenery obstruction, fuel consumption, vehicle-specific limitations (turning radius, ascent gradient, size) *etc* (Ait Saadi et al., 2022; Hildemann & Verstegen, 2023). It is also possible to distinguish between "soft" objectives that minimise an undesirable metric, and "hard" imperatives that are restricted outright. For example, it may be preferred to traverse less densely populated areas, but highly populated areas are not off limits (soft). On the other hand, it is physically impossible to intersect a building, which makes it a critical and unwavering objective (hard).

2.2 Systematic Literature Review

2.2.1 PRISMA Method

In order to ensure an exhaustive and reproducible literature review, to the extent that this is possible, a systematic literature review (SLR) was conducted according to the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) guidelines (Page et al., 2021). The method starts with a research repository search using a set of keywords in a constant expression.

The focus of the SLR is trajectory optimisation in the context of 3D navigation, using artificial intelligence techniques. Therefore, the search criteria includes two title-specific requirements and one more generic (title, abstract or keywords) requirement. Firstly, results

must contain “trajectory”, “path” or “route” as well as one of the renditions of “optimisation” in the British or American spelling. To exclude outdated references the results are also limited to papers published after 2016. The Scopus research repository is considered and the final search expression was used in October 2022 and is shown below:

```
TITLE ( "trajectory OR "path" OR "route" ) AND TITLE ( "optimisation"
OR "optimization" OR "optimising" OR "optimizing" OR "optimise" OR
"optimize" ) AND TITLE-ABS-KEY ( "artificial intelligence" OR
"machine learning" ) AND PUBYEAR > 2016
```

Exclusion criteria were chosen to maximise the relevance of retained articles. First, optimisation techniques that exclusively involve routing in graphs are identified and excluded. This includes any subdivision of the search space into a non-continuous medium. Second, articles that do not explicitly involve path optimisation are identified and excluded. Thirdly, techniques that do not require obstacle avoidance as part of the pathfinding process are excluded. Trajectory optimisation in two dimensions is included since some of the techniques may be relevant in 3D space.

2.2.2 PRISMA Results

The results of the PRISMA systematic literature review are shown in the flow diagram in Fig 3.

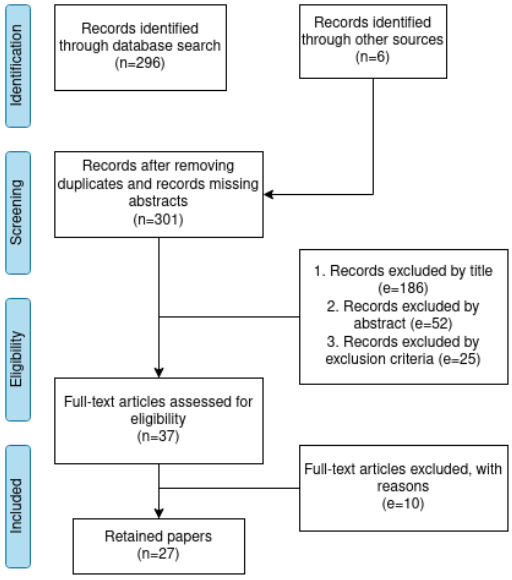


Figure 3: PRISMA flowchart

The Scopus search returned 296 results, with a further 6 sourced manually from other repositories. In these 302 records there were no duplicates, and a single record was excluded for lacking an abstract. 301 records were thus assessed for eligibility.

The majority of exclusions by title were on the grounds of graph-based optimisation being the focus of the research. Synonyms for this domain are "combinatorial optimisation", "graph search" *etc.* This includes research into air and road traffic management, and robotic pathfinding in a grid-based environment. These techniques focus on node-to-node topological optimisation, or optimising the node order in a multiple-target setting. Some of the optimisation methods limited to graphs include Swarm Intelligence (SI) techniques such as ACO and PSO, Artificial Bee Colony (ABC), Cuckoo Search (CS) and Invasive Weed Optimisation (IWO). Many of these articles were returned by the search due to routing and pathfinding often being used interchangeably.

Another prolific area of research is trajectory planning for serial manipulators. However, these are excluded because the trajectory is solely a result of the orientation of the joints, and the base and total length are constant. Other exclusions were for single-stage to orbit spaceplane trajectory planning, lunar lander hopping trajectories and interplanetary trajectories, which all involve parameterisation of a starting configuration that will result in a desired final configuration, after considering external effects.

Where the title was not sufficiently explicit to warrant exclusion, the record's abstract had to be scrutinised for eligibility. In many cases the increased detail revealed the research's non-applicability and a majority of rejections were again due to the research being limited to graph optimisation in a graph.

With 239 records already excluded by title and abstract, the remaining 62 were studied to assess relevance by consulting the full text. This process led to another 25 exclusions, leaving 37 articles for consideration in the literature review.

Of the 301 articles 106 were excluded on the basis of being routing methods navigating through an existing network, as opposed to path finding in an unconfined search space (regardless of dimensionality). Another 25 articles applied exclusively to routing within electronic networks. After the eligibility phase, articles are filtered a final time for valid but unanticipated reasons. Ten articles were thus excluded on the grounds of relevance, leaving 27 retained articles.

2.2.3 PRISMA Discussion

Figure 4 shows the keyword co-occurrence of the 296 returned articles. Out of 868 keywords the ones with 4 or more occurrences are visualised. From this, the popularity of swarm intelligence techniques such as PSO and especially ACO is immediately evident. *Machine learning* is the second most common keyword and is followed by *artificial intelligence*, both of which frequently occur alongside *genetic algorithm*. *Energy consumption* also appears here, testifying to its importance as an optimisation factor. Mobile robots are the most common application used as a keyword, in 9 articles.

Twenty retained articles (74%) are constrained to 2D space. Two use a cost surface as a third dimension (Balogun et al., 2017; García et al., 2022), and one optimises the x and z dimension (Pérez-Cutiño et al., 2022). Applications range from paths for UAVs (13 records), mobile robots (7 records), USVs (2 records) and spacecraft (2 records) to UUVs, ships and submarine pipelines (1 record each).

PSO is the most popular technique for stochastic optimisation, used in 6 articles. Mokhtari (2022) generated 3D paths by mapping a B-spline onto PSO-generated points, and validated

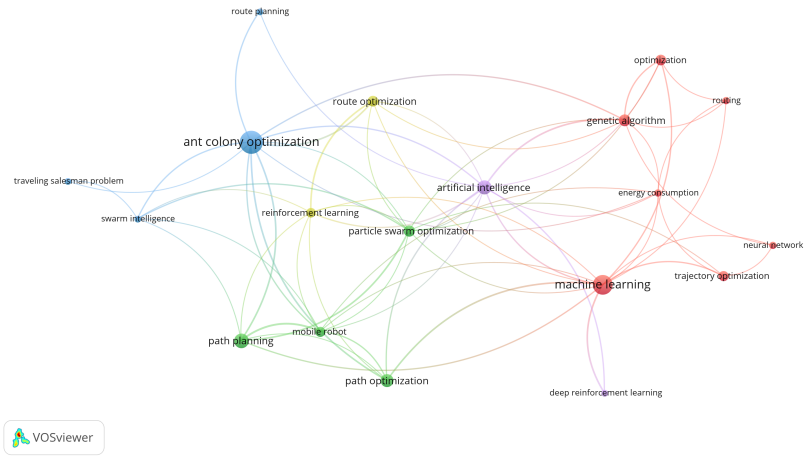


Figure 4: Keyword co-occurrence network visualisation

solution candidates by measuring their distance from obstacles. Oultiligh et al. (2020) compared Dijkstra, Voronoï, PSO and Fuzzy Logic and concluded that PSO is superior in that it yields the shortest possible path, whereas Dijkstra struggles in complex environments and Voronoï guarantees obstacle avoidance but cannot yield an optimal path. Other common techniques include deep learning, used by Qie et al. (2019) to plan paths for multiple UAVs while avoiding threat areas.

Only eight articles discuss 3-dimensional trajectory optimisation and of these, four additionally consider obstacle avoidance. Zhang and Duan (2015) use Predator-Prey Pigeon-Inspired Optimisation (PPPIO) in a dynamic 3D environment, effectively incorporating a fourth dimension. Their novel algorithm is not only an improvement over PIO but was also shown to outperform PSO and differential evolution (DE).

Ma et al. (2020) and Sundaran (2018) emphasise the importance of path smoothness for ease of navigation, citing the turning abilities of different vehicles. They use Bézier and B-spline curve generation respectively to obtain smoothed paths, whereas Zhang and Duan (2015) use a dynamically feasible path smoothing strategy called κ-trajectories.

Evolutionary algorithms were used in three papers: GA applied to mobile robots in 2D (Mane & Vhanale, 2019) and UAVs in 3D (Hildemann & Verstegen, 2023), and GP applied to USVs in 2D (Jing et al., 2022). Thus the literature review suggests that potential exists for GP applications in 3-dimensional space with obstacle avoidance.

The literature review findings shows that a great deal of research has already been invested into trajectory optimisation. However, it is overwhelmingly 2D-constrained with only a tiny minority of articles venturing into the third dimension. This represents a plethora of potential navigational applications that have yet to be addressed. The majority of published research uses swarm intelligence based EA optimisation, and GP based approaches are severely underrepresented. This leads to the conclusion that a 3D GP trajectory optimisation is a novel approach worthy of investigation.

3 Methodology

The literature review findings established the scientific background for the rest of the study. The proposed methodology to support this research is guided by the achievements of existing research as well as the shortcomings thereof described earlier in Section 2.2.2. GP is chosen over GA due to its more compact representation and speedier computation. The problem is extended to three dimensions in order to broaden the potential benefits of its application. The encoding method of the solution geometries and the transformation into geographical space are also novel, with no similar methods having been previously developed. The proposed methodology therefore represents a unique approach to 3D trajectory optimisation that mainly builds upon the findings of Hildemann and Verstegen (2023), a co-supervisor of this thesis.

In the simplest terms, the goal is to generate an optimal path from a given start and end point in three dimensional space without intersecting any restricted areas. These barriers and two points comprise the geographical inputs. The only other input is the set of parameters which, to varying extents, affect the outcome and the performance of the optimisation. Adjusting these input parameters carefully can effectively calibrate the optimisation model to yield better or quicker results. As output we expect a 3D linestring connecting the origin and destination points in the most cost-effective manner, avoiding all obstacles.

3.1 Study Area

A section of New York City has been selected as the primary test case. The reason is threefold: firstly, the study area has shown itself to be appropriate and popular for air taxi studies (Hildemann & Verstegen, 2023); secondly, it allows a direct comparison of the results of said prior research with the results of this work, yielding greater insight into its academic value; and lastly, the data is freely available and has been largely preprocessed already. The map in Fig 5 shows the study area along with the pathing result of Hildemann and Verstegen (2023), which this research aims to improve upon.

Clove Lakes Scenario Cunningham Park Scenario

Figure 5: New York City restricted flight areas

The urban area of New York is rife with flight obstacles due to flight regulations (restricted airspace) and public disturbance (protected airspace). Ground and sea level are the obvious lower flight limits. However, flights are also restricted to a minimum height of 152.4 m above buildings, and due to the density of buildings in New York this is the *de facto* lower limit. The upper limit imposed by the Federal Aviation Administration (FAA) due to plane air

traffic is 213.36 m. Protected areas include schools, cemeteries, recreational areas *etc.* and have vertical and horizontal no-flight buffer zones as shown in Table 1:

Land use	**Vertical Restriction**	**Horizontal Restriction**
Airport	600	7620
Hospitality	300	300
University	200	300
Park	300	100
Cemetery	300	100
Recreational Area	300	100
Rooftop	152.4	-

Table 1: Vertical and horizontal restrictions in meters (Hildemann & Verstegen, 2023)

3.2 Data Preparation

The study area has been partially prepared by Hildemann and Verstegen (2023) and an ESRI file Geodatabase (available at https://github.com/mohildemann/3D-Flight-Route-Optimization) is ready for use for some test cases. Public city data, made available by Open Data NY, OpenStreetMap and the Federal Flight Agency, has been processed to yield barriers. Buffer zones have been created around restricted and protected land use areas. These were then extruded by the appropriate flight height clearance limits to yield 3-dimensional polygons of restricted areas. A minimum forbidden airspace exists above all rooftops, and an upper limit is in effect universally. The ground forms the absolute minimum constraining flight altitude.

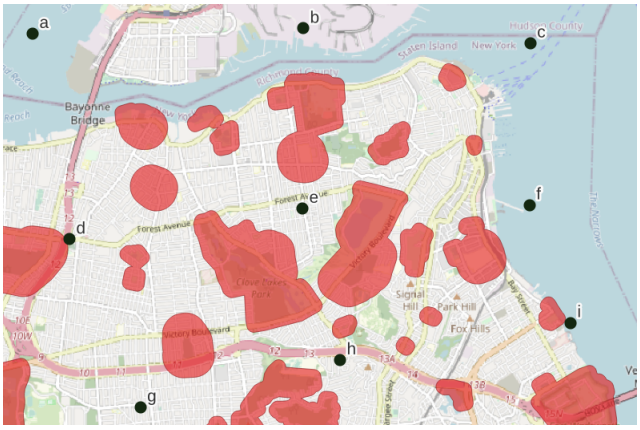


Figure 6: Clove Lakes dataset showing points

A test area containing 17 barrier polygons around Clove Lakes Park in northern Staten Island was prepared by manually selecting points to be used as start and end points, with the intention of providing various combinations of obstacles between pairs of points in order to assess the GP performance. Another scenario was created that includes the whole Staten Island (42 polygons). The hardest route to find is in the third scenario, in the same area near Cunningham Park used by Hildemann and Verstegen (2023). The universal height

limit was varied or removed to assess the optimisation's response to changing availability of the z-dimension.

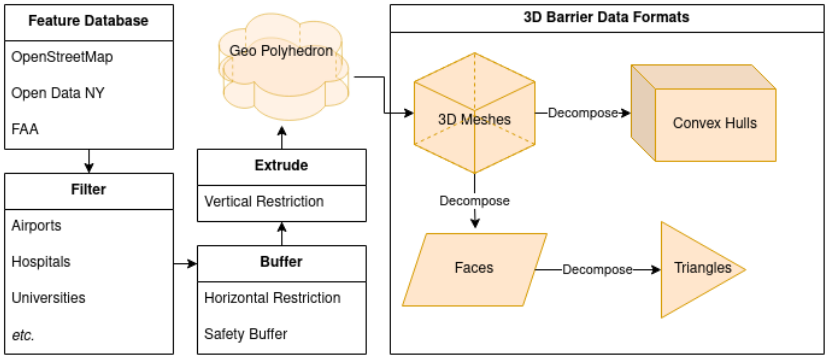


Figure 7: Data preparation flowchart

Three-dimensional intersection algorithms using only polyhedrons that are convex hulls are significantly more efficient than algorithms using more complex concave meshes. To use these routines it is therefore necessary to prepare additional preprocessed datasets, consisting of only geometric primitives such as faces, convex hulls or triangulated meshes. How this is done depends on the input format, but generally the 3D barriers are deconstructed into their parts. Triangles are simply faces limited to three vertices, and convex hulls are simply 3D meshes without concavities. Figure 7 shows the data preparation process to produce different 3D geometry formats for use in different validation methods. The type of intersection calculation varies with the barrier geometry: for concave hulls, a barrier is intersected if any of the vertices of the 3D line are contained within the hull; for faces and triangles, a barrier is intersected if any of the 3D line segments intersects any of the 2D barrier components.

3.3 Evolutionary Computation Framework

Distributed Evolutionary Algorithms in Python (DEAP) is a framework for developing evolutionary computation techniques in the Python programming language (Fortin et al., 2012). The code and data used in this thesis is available at https://github.com/andre-kotze/gp-trajec. In DEAP, classes exist to typify function trees (*syntax trees*) as individuals, consisting of operators and terminals (primitives) that comprise the primitive set (Poli, Langdon, et al., 2008). A `base.Toolbox()` instance is created to manage the evolutionary operators and register functions used in the optimisation. Here the creation, selection, crossover and mutation methods are specified, and limits are defined as decorators.

Figure 8 shows the generalised GP flowchart. During initialisation, a starting population is randomly generated according to user-set parameters. These candidate solutions are then validated by checking whether their intersection with the barriers is an empty set. Valid solutions are then evaluated *i.e.* their fitness is quantified. A cost function is called that measures an individual's aptness as a solution, according to the optimisation criteria, and the population is subsequently ranked by fitness. With all the fitnesses known, the best-performing individual is compared to the stopping criteria. If this solution is sufficiently apt, the routine terminates and the optimisation is complete. If not, the population is varied

by mating pairs of individuals or randomly mutating genes in an individual. Individuals with greater fitness have a greater probability of mating, mutating or entering the next generation unchanged. After this variation, the offspring generated becomes the starting population of the next generation, and the process repeats.

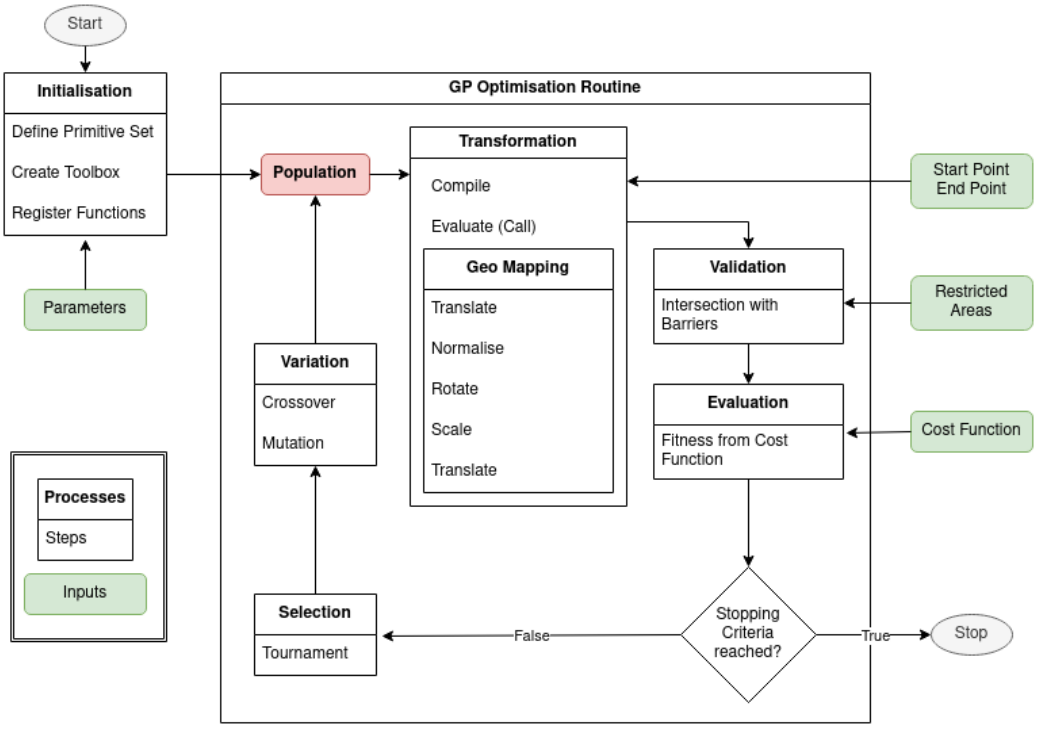


Figure 8: Genetic Programming methodology flowchart

Variation algorithms are used to vary the parent population to produce the offspring, via crossover of genes, mutation or both. In DEAP, the `VarAnd` variation algorithm is so called because it applies crossover and mutation. The operators are not applied systematically. Rather, an initial iteration over the population mates consecutive individuals according to the crossover probability `cxpb` with the resulting individuals replacing their parents. In this work the `tools.cxOnePoint` algorithm is used for crossover, which only swaps a sub-branch from each tree at one point and therefore the resulting trees have the length that the other had. A second iteration mutates individuals according to the mutation probability `mutpb` with the mutated individuals replacing their unmutated versions. The mutation method used is `tools.mutUniform` which selects a random point in the tree and replaces the sub-branch stemming from this point with a newly generated tree. Thus the offspring consists of new individuals that are crossed-over, mutated, crossed-over and then mutated, or unchanged.

After evaluating the fitnesses of the population, selection is done using one of the tournament-based selection methods `tools.selTournament` or `tools.selDoubleTournament`, which iteratively selects the best solution from a randomly selected subset of individuals. The

double tournament has a second stage where solution size is considered instead of fitness, and has been shown to effectively control bloat by preferring smaller individuals (Luke & Panait, 2002). The selection method can take multiple objectives into account to a limited extent, since the fitness is derived from the cost function where different fitness metrics have different weights.

3.3.1 Parameterisation

DEAP allows extensive customisation of its internal algorithms, as well as facilitating a high level of parameterisation. Tuning these settings via experimentation can improve the optimisation method's performance by calibrating it to specific use cases, for example where greater precision or higher speed is required, or where complex scenarios require larger trees and more iterations. Default values exist for all parameters and are given in Table 2.

For this research an extensive and accessible parameter dictionary is utilised and categorises the parameters into 5 configuration categories based on the aspect of the model that they affect:

`dataset`	Spatial input data to use
`defaults`	GP and optimisation settings
`validation`	Validation-specific settings
`logging`	Logging-related settings
`visualisation`	Plotting-related settings

When the starting population is generated, its size is dictated by the `pop_size` parameter. Population size remains constant throughout the evolutionary process. Population size directly affects diversity and higher values have greater search potential. However, this also incurs a significant performance cost. The size of individuals is limited in length and height by the `max_length` and `max_height` parameters respectively, which limits the number of nodes in a primitive tree as well as the depth (maximum distance from the top of the tree). For example, the tree in Figure 2 has a length of 10 and a height/depth of 4. Similar to population size, larger trees encapsulate more diversity and complexity, but are detrimental to performance. The `init_min` and `init_max` parameters control the minimum and maximum length of newly initialised trees, which will have a random length within this range. Higher values can yield more complex solutions earlier on in the evolution.

Parameter	Type	Default	Description
`ngen`	`int`	`250`	Number of generations
`mutpb`	`float`	`0.1`	Probability of mutating an individual
`cxpb`	`float`	`0.5`	Probability of mating two individuals
`nsegs`	`int`	`100`	Number of line segments in the 3D path
`patience`	`int`	`100`	Number of generations to wait for improvements
`max_height`	`int`	`17`	Size limit per individual, by number of levels
`max_length`	`int`	`80`	Size limit per individual, by length
`pop_size`	`int`	`1000`	Population size
`elitism`	`Bool`	`True`	Whether to implement elitism
`hof_size`	`int`	`10`	Number of elite individuals to preserve
`adaptive_mode`	`Bool`	`True`	Whether to quantify intersections

Table 2: Genetic programming parameters

The `ngen` parameter sets the number of planned generations for the optimisation to evolve

through. This is the ultimate stopping criterion at which point the program will terminate regardless of the quality of the optimisation achieved. If no improvement is seen over a significant number of generations, it can be assumed that the optimisation is practically complete and that further processing is unnecessary. This number is set by the `patience` parameter.

During variation, an offspring population is produced from the parent population by randomly applying crossover and mutation operations. Individuals are looped over sequentially and two consecutive individuals are crossed over according to the probability `cxpb`. After the mated individuals have been replaced by their two children, a second loop applies the mutation operation according to the probability `mutpb`. Higher values lead to greater differences between parent and offspring populations.

When deriving a curve of finite length from a function tree, it is important to consider the interval in which to select the curve. The lower and upper x-axis values are thus given by the `start` and `end` parameters. The granularity of the curve *i.e.* the number of line vertices, is given by `nsegs` and is one more than the number of line segments. More vertices yield a more resolved linestring but significantly affect processing speed. It is conceptually equivalent to the chromosome length in studies where lines are traditionally encoded.

Top individuals can be selected and saved into the "Hall of Fame (HoF)", which is updated each generation after fitnesses have been updated. The HoF preserves the set of best solutions from the whole optimisation and individuals in the HoF can only be replaced by fitter individuals from subsequent generations. The number of solutions preserved is controlled by the `hof_size` parameter. The `elitism` parameter determines whether elitism is implemented, and the number of elite individuals carried over is equal to the HoF size.

To toggle between selection by tournament or double tournament, the `dbl_tourn` parameter may be adjusted. The `tournsize` parameter determines the number of individuals participating in each tournament. Double tournament takes two additional parameters. `parsimony_size` defines the weight of the solution size in the second part of the tournament, and must be a real number in the range [1,2]. `fitness_first` determines whether the fitness or the size stage of the tournament is held first.

3.3.2 Solution Anatomy

Two degrees of freedom in the pseudo-y and pseudo-z dimension of the solution space allow individuals to develop increased complexity and greater likelihood of finding valid trajectories in complex obstacle-rich scenarios. These can be thought of as lateral and vertical deviation from the straight line (pseudo-x) connecting the origin and destination point. To yield these truly 3-dimensional lines an additional input argument is added to the primitive set so that the function tree accepts two arguments. These can then be individually evaluated by setting the other value to 0 (or any other arbitrary value) to produce two distinct outputs.

Smoothness and solution granularity is an important factor for navigability of the trajectory due to the differing turning radii and attainable ascent gradients of UAVs. It is also beneficial in terms of energy cost to navigate along smoother routes (Ma et al., 2020). Hildemann and Verstegen (2023) used a set of vertices (3D points) with a corresponding set of smoothed lines connecting them, whereas Ma et al. (2020) used a set of control points to constrain a Bézier curve. In this work, the smoothness derives from the curve of the functions represented by

the genetic programs, and the granularity depends directly on the ratio of the trajectory length and the `nsegs` parameter.

3.3.3 Bloating

As a result of continuous variation over generations, tree size generally increases disproportionately to fitness. This phenomenon is known as bloat (Poli, Langdon, et al., 2008; Whigham & Dick, 2009) and can drastically reduce the optimisation performance. Four methods of bloat control are implemented in this work: a global hard limit on the size of trees, double tournament selection, size as part of the cost function, and an elitist strategy. The global limit suggested by Koza (1992) is 17, but allows for enormous programs up to $131,072$ (2^{17}) in length. Limits of 8, 12 and 17 are used in this study in combination with other bloat control methods.

3.3.4 Elitism

While the average fitness generally improves over generations, high-fitness "elite" individuals may be lost if their offspring are less fit. The stochastic nature of the optimisation means that it is just as likely for offspring to be better as it is for them to be worse. Elitism is implemented to prevent any loss of fitness by preserving elite individuals unchanged for the next generation.

At its least influential, elitism prevents any deterioration of the population's maximum fitness. It is therefore guaranteed that the best individual ever will be in the final population. Elitism also has the potential to accelerate the optimisation by increasing the number of pairing instances that high-fitness individuals participate in, and even counteracting the propensity for bloat (Poli, McPhee, et al., 2008a; Poli, McPhee, et al., 2008b). In this work, elitism is enabled by default, but can be disabled. The number of elite individuals retained per generation is determined by the `hof_size` parameter, and the *elite fraction* is defined as the ratio of `hof_size` to `pop_size`. Increasing the elite fraction can prove beneficial, but also carries a risk of premature convergence due to the corresponding cost to diversity.

3.3.5 Stopping Criteria

Evolution will continue until one of the stopping criteria is reached. The simplest stopping criterion is the end of the specified number of generations. If the patience parameter is set, the evolution will terminate if no improvement is seen in the given number of generations. Alternatively, termination can be triggered once the first valid solution is found (useful for highly complicated environments), or once the fitness reaches a specified threshold (useful for time-critical scenarios).

3.4 Solution Transformation

The individual validation procedure starts with decoding the chromosome and compiling the program into a callable function. This function is then plotted in 3D space in an arbitrary interval to yield a finite path between two points (Figure 9). In order to validate this path, it is transformed to map onto the problem interval in geographic space. Once this path connects the input start and end points, its suitability can be assessed according to the chosen criteria. For the transformation procedure, the start point, end point and number of segments are required. The process is outlined below:

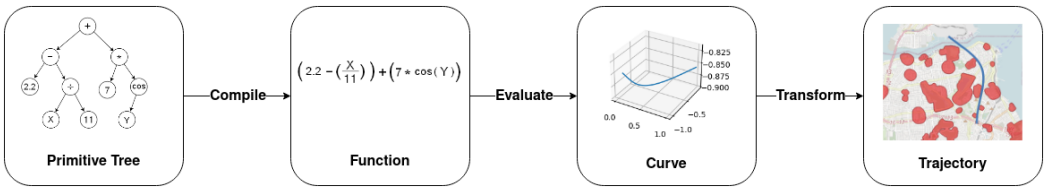


Figure 9: Solution transformation procedure

1. translate to origin
2. normalise
3. rotate about origin
4. scale to geographic scale
5. translate to destination

To facilitate rotation about the origin, and to anchor the start point, the solution curve is translated to the origin. The displacement of the start point coordinates is used as the translation magnitude:

$$\text{Translation:} \quad T_{\sigma x,\sigma y,\sigma z} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \sigma x & \sigma y & \sigma z & 1 \end{pmatrix}$$

The next step is normalisation of the vector, which is simply dividing the vector by its magnitude. Here we consider the direct displacement from the start point to the end point (*i.e.* the net displacement of the path) as the vector:

$$\text{Normalisation:} \quad \hat{u} = \frac{u}{|u|}$$

After normalising we are left with a unit vector that we want to align with the geographic displacement by rotation. The simplest method is to measure the angles between the two vectors in each plane separately, using arctan2, and negate the difference. The vector can then be rotated through this angle using Euler rotation, in each plane separately:

$$R_{z,\theta} = \begin{pmatrix} cos(\theta) & sin(\theta) & 0 & 0 \\ -sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_{y,\theta} = \begin{pmatrix} -sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ cos(\theta) & sin(\theta) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_{x,\theta} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ cos(\theta) & sin(\theta) & 0 & 0 \\ -sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

However, this method introduces error due to gimbal lock, a well-known limitation in 3D rotation using Euler angles (Mansur et al., 2020). Gimbal lock occurs when two axes align during rotation in 3 dimensions and cause the rotational system to lose a degree of freedom. Instead of three consecutive matrix calculations for the three axes, quaternions can be used to rotate the vector in a single direction a single instance. This translates to a faster and more precise rotation operation.

Quaternions (versors) provide a convenient way of expressing and applying rotation in 3D space using imaginary numbers. They have four components i, j, k, w: where w is the magnitude of the rotation and i, j, k are a unit vector in imaginary space, representing the axis of rotation. Unlike rotation matrices they are not susceptible to gimbal lock, and since they have only four components and are applied only once, they are also computationally more efficient than matrices (Huynh, 2009).

In order to align the solution curve and the geographic interval, both must first be translated to the origin and normalised as vectors *i.e.* only the start- and end-points are used in this stage, to create a pair of unit vectors. The quaternion initialisation requires two arguments: the overall axis of rotation in *ijk* space (equal to the cross product of the two unit vectors) and the angle of rotation (scalar, equal to the dot product of the two unit vectors). Once the quaternion is created, the same magnitude of rotation is applied to each vertex of the solution line to align it with the geographic interval.

$$\text{Rotation:} \quad R = \begin{pmatrix} 1-2s(q_j^2+q_k^2) & 2s(q_iq_j-q_kq_r) & 2s(q_iq_k+q_jq_r) \\ 2s(q_iq_j+q_kq_r) & 1-2s(q_i^2+q_k^2) & 2s(q_jq_k-q_iq_r) \\ 2s(q_iq_k-q_jq_r) & 2s(q_jq_k+q_iq_r) & 1-2s(q_i^2+q_j^2) \end{pmatrix}$$

The penultimate step is scaling the curve to the size of the geographic interval. For this, a simple scaling matrix is used, with the scale factor k equal to the magnitude of the geographic interval divided by that of the solution curve.

$$\text{Scaling:} \quad S_k = \begin{pmatrix} k & 0 & 0 & 0 \\ 0 & k & 0 & 0 \\ 0 & 0 & k & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Finally, the result is translated from the origin to map onto the geographic space. Once there, it can be validated and evaluated.

3.5 Path Validation

In this stage, the geographic path is plotted in the same space as the 3D barriers and validated by checking whether any intersection exists between it and the barriers. Several approaches can be used in the validation process, varying in accuracy and performance. Performance is affected by the intersection algorithm used as well as the geometry representation: the solution can be considered as a collection of line segments (high performance

cost) or vertices (low cost). Likewise, the barriers can be considered as 3D meshes (high cost), collections of concave hulls (medium cost) or collections of 3D polygons with a height attribute (low cost). Another possibility is using 2D polygons with a height attribute. This has a low performance cost but limits the third dimension to a single range per polygon.

Accuracy is affected similarly by geometric representation. A path's vertices might not intersect a barrier, although its segments do (Hildemann & Verstegen, 2023). More intricate geometries can be represented with 3D meshes than with convex hulls or polygons.

In this work, two main approaches to 3D intersection are considered. The first is a simplified two-and-a-half-dimensional (2.5D) geographic intersection, where the x-y intersecting geometry (if it exists) is evaluated separately in the z-dimension. The restricted height range is taken from the barrier data attributes. Although fast and applicable to many real scenarios, this approach is limited in its representation of complex 3D environments.

The second approach is a fully 3-dimensional intersection using convex hulls and points. It is therefore limited to convex geometries (although concave shapes can be decomposed is an additional preprocessing step) and is susceptible to omission errors due to its use of the route vertices instead of line segments.

Invalid paths are either deleted or their fitness severely penalised, depending on the model configuration. Deleting invalid individuals ensures that the population only contains viable solutions that will continue to be optimised. However, as the complexity of a scenario increases the likelihood of initialising a valid individual decreases, and it is not unusual for no valid solutions to ever be randomly initialised. Penalising invalid routes indicates to the model the undesirability of such routes, but allows the evolution to continue by allowing the individual to be varied.

3.6 Fitness Evaluation

Following validation, paths are evaluated for fitness, primarily by measure of length. Additional fitness factors, or optimisation objectives, are incorporated in the cost function. In this way a ranking within the generation is established for the selection operation to operate on.

An individual's fitness is derived directly from what is returned by the cost function. The first consideration is the total path length, the traditional subject of path optimisation. Other metrics with adjustable weight include solution size and path variability in the z-dimension (this has a direct and significant energy cost). Path length is not measured directly in 3D, as the cost of moving through the z-dimension is not equivalent to moving parallel to the surface. Length is measured in the x-y plane and variation in the third dimension is measured separately, as absolute cumulative displacement. Thus vertical movement tends to become minimised and the ratio between these measurements can be adjusted.

Metric	**Weight**	**Identifier**	**Description**
Length	0.85	l	Path length
Altitude variation	0.1	z	Cumulative absolute z-dimensional displacement
Size	0.05	s	Chromosome length * height

Table 3: Variables considered in the cost function

4 Results

In two dimensions, the GP method is robust against complex obstacle environments and usable results are obtained in a relatively short time. Results in 3D may be quicker in some cases, if a simple route over nearby obstacles is available, although the time per evaluation is greater. Accounting for the number of generations, the duration of 2- and 3-dimensional optimisation is mainly affected by population size, geometry size (number of vertices), tree size and scenario complexity. The quality of the optimisation is mainly affected by the broadness of the search, which depends on variation and selection (including elitism) settings.

Figure 10 shows two trajectories evolved to connect points b and bc. The 2D trajectory is clamped to the ground and must navigate between and around the barriers, whereas the 3D trajectory has the added ability of passing over barriers whenever doing so would shorten the route. Due to the innate smoothness of generated trajectories and the weight of height minimisation, the 3D route does not go over the obstacles nearest to the start and end points. Instead it circumvents them, similar to the 2D route, and only directly passes above the obstacles near the centre of the trajectory.

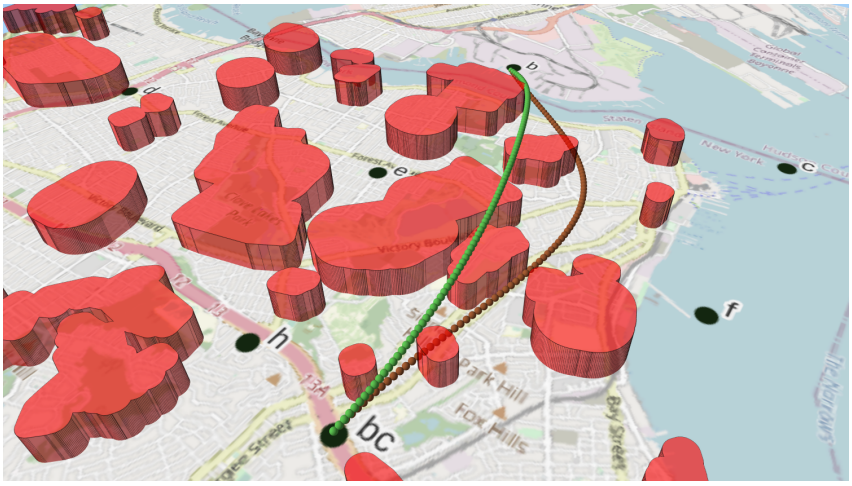
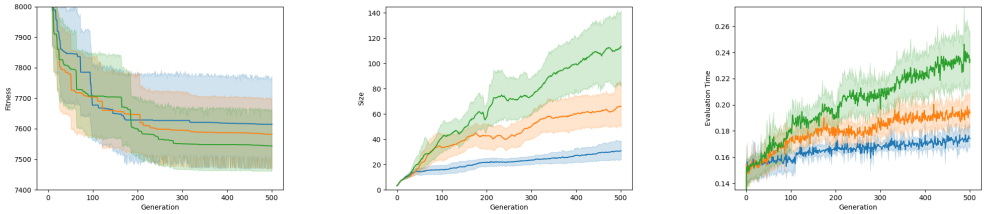


Figure 10: Comparison of 2D (brown) and 3D (green) trajectories.

Varying the maximum tree height shows a clear advantage for higher values in terms of fitness (Figure 11). Significant variability of the 95% confidence interval continues into later generations, which hints at local minima being encountered, and the propensity for this is directly correlated with the tree height limit. Although evaluation time increases proportionally to tree size, the increased diversity of larger trees allows them to outperform more restricted trees. In the initial 100 generations the middle value is superior, before being mostly superseded by the smallest trees up until 190 generations, and the optimisation is finally dominated by the largest trees. This suggests that parameters have distinct advantages at different stages of the evolution. Lower height limits can yield quicker solutions, but do so inconsistently. Higher limits ensure a finer convergence, at the expense of solution size and optimisation speed.



Fitness Size Evaluation Time

Figure 11: Effect on fitness, size and evaluation time for maximum tree height 8 (blue), 12 (orange) and 17 (green).

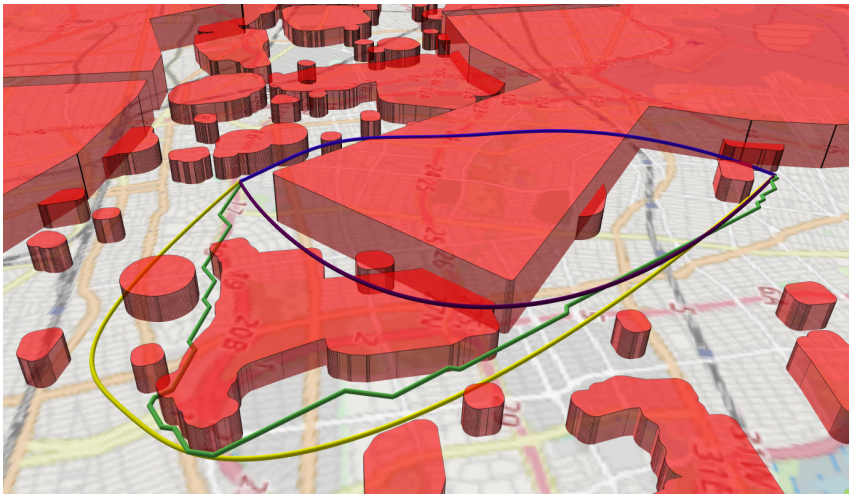


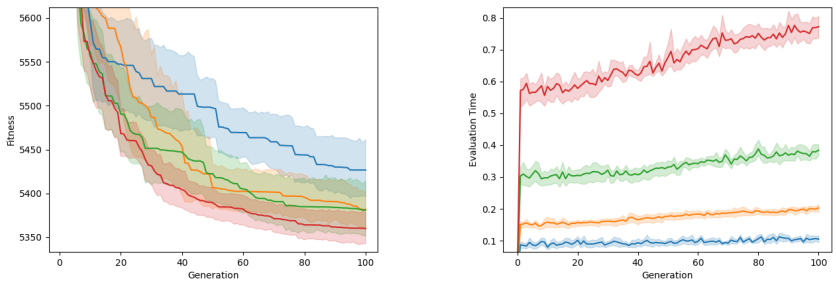
Figure 12: Trajectories with no height limit (blue), a 500 m limit (purple), and the actual 213 m limit (yellow). Compare with Hildemann and Verstegen (2023) (green).

Figure 12 shows the trajectories generated after 1000 generations, with quantified intersections, compared to the results of Hildemann and Verstegen (2023). Most striking is the difference in length and smoothness. Under identical constraints (height limitation 213 m) the GP line is 12,469 m in length, compared to 12,513 m. This is a minuscule reduction in length, however the gradual curvature in the GP solution presents a significant reduction in flight time and energy cost due to the increased acceleration and deceleration required to navigate sharp changes in heading.

Varying the global height limit also shows how the algorithm is able to adapt to different scenarios. In the real world a 213 m flight height limit exists, which forces the optimisation to navigate around the 300 m-high restricted areas. The 500 m limit represents a situation where the airport zone is completely impassable, but the 300 m high zone over the park can be flown over. With no global height limit, the optimal route goes directly over the airport zone.

4.1 Initialisation and Convergence

Population size is one of the two major numerical parameters of GP (Koza, 1992). Poli, Langdon, et al. (2008) recommend using the highest value that can be gracefully handled by the system in use. Figure 13 shows the outcomes of four different values for population size with all other parameters kept constant. It shows a clear improvement in optimisation efficiency but a direct cost to evaluation time, as expected. Over 10 optimisations the larger populations converge more finely in the minimum fitness, but diverge more in the evaluation time.



Fitness Evaluation time

Figure 13: Optimisation and duration for population sizes 300 (blue), 500 (orange), 1000 (green) and 2000 (red).

Considering the stochastic nature of the initialisation, variation and selection operations, it is not surprising that optimisations with the same inputs can vary significantly. In order to visualise the convergence of the evolutionary process, multiple optimisations for the same problem are computed, varying the random seed state each time. The result is a relatively high diversity in the fitness values in earlier generations, which eventually converges as the objective is reached. The 95% confidence intervals in Figure 14 show that optimisations with different starting states converge to a near-optimal solution after 50 generations for the scenario shown in Fig. 14 (a) ($CoV = 0.2\%$), and 20 generations for the scenario in Fig 14 (b) ($CoV = 1.1\%$).

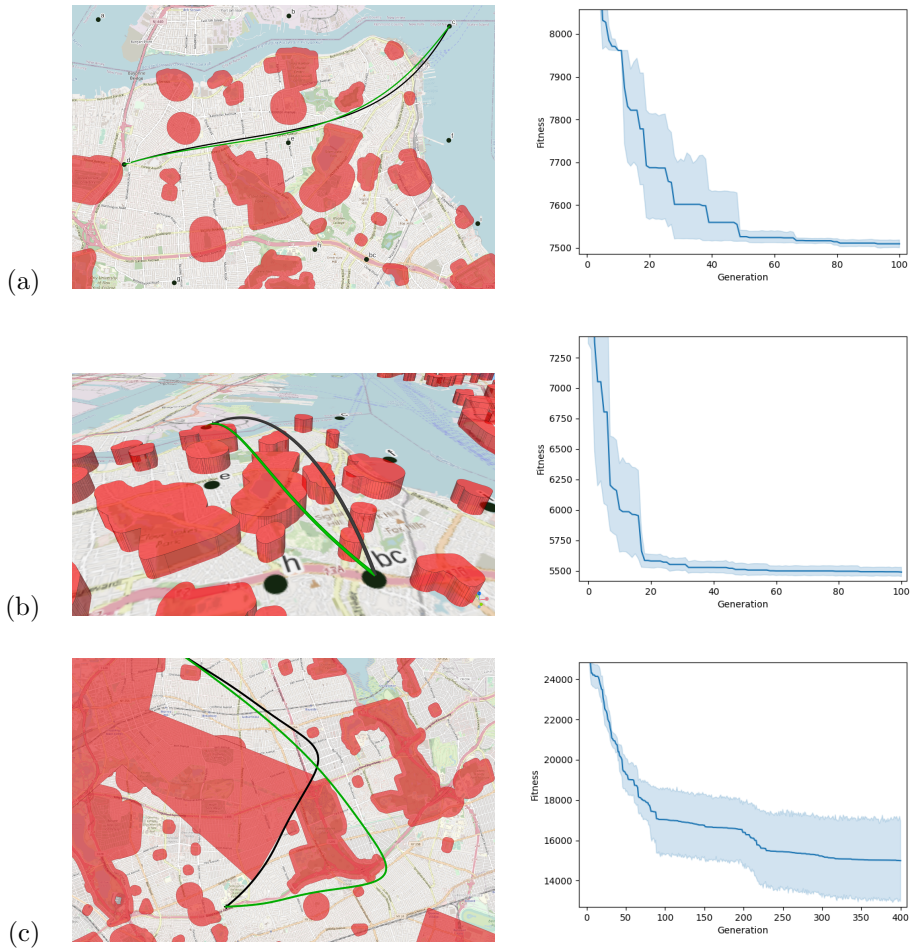


Figure 14: Convergence in three scenarios, showing the 95% confidence interval. Map trajectories correspond to the best (green) and worst (red) outcomes of 10 optimisations.

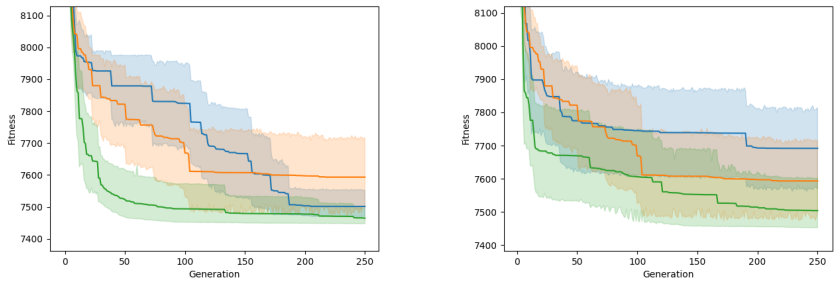
In the third scenario (c), the optimisation fails to converge after 400 generations ($CoV = 21.9\%$). Due to the complexity of this scenario the search is vulnerable to local optima, and the initial state has a significant effect on the final outcome. The minimum fitness after 400 generations ranges from 11979 m to 25434 m

4.2 Variation and Selection

Crossover and mutation directly affect the diversity of the population, which is effectively the search radius. Additionally, crossover can produce more specialised offspring by combining the beneficial traits of both parents. Figure 15 shows how optimisation performance increases with a higher probability of both crossover and mutation. This illustrates how larger values allow for a quicker and broader search and may aid in avoiding or escap-

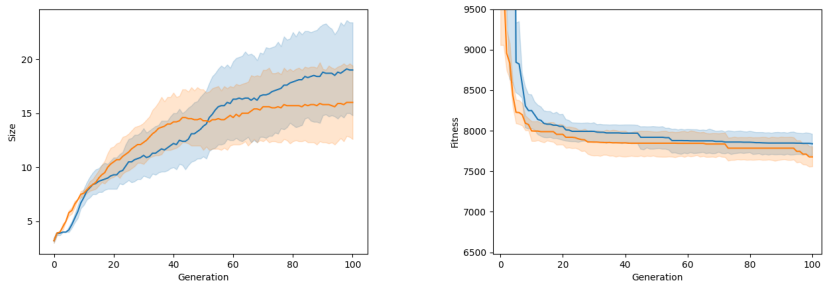
ing local minima. Increasing these values also increases the processing time because more evaluations are required for the higher number of altered offspring.

Considering size in the selection process, as an additional tournament stage, means that smaller solutions are preferred if fitness is similar. The results in Figure 16 show the reduced average size in the double tournament experiment, but also show a slight improvement in the optimisation.



Crossover Mutation

Figure 15: Optimisation convergence for crossover probabilities of 0.2 (blue), 0.5 (orange) and 0.9 (green), and mutation probabilities of 0.05 (blue), 0.1 (orange) and 0.25 (green).

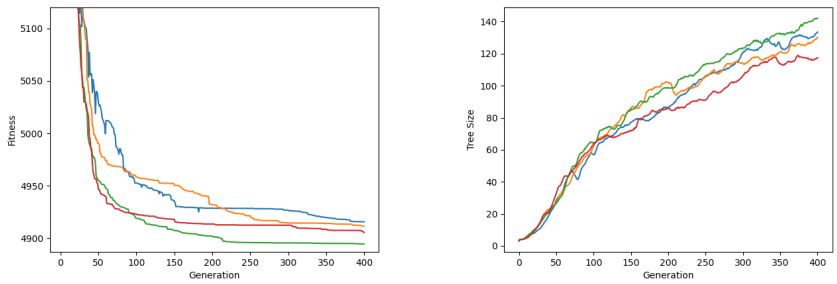


Size Fitness

Figure 16: Size and fitness using tournament (blue) and double tournament (orange) selection.

4.3 Elitism

Enabling elitism can improve convergence speed due to its preservation of the best individuals. The best fitness in the population is therefore protected from degrading with continued evolution. Optimisation results show that the minor loss of diversity is offset by the increase in speed and quality, depending on the size of the elite fraction (see Figure 17). However, the effect is not proportional: after 100 generations, elitism with 3% elite fraction outperforms 10%. Elitism with 10% of individuals preserved performs the best in terms of optimisation speed and solution size.



Fitness Tree size

Figure 17: Elitism effect with elite fraction 0% (blue) 1% (orange), 3% (green), and 10% (red). Average of 10 optimisation runs.

Effective bloat control using elitism is well-documented (Poli, McPhee, et al., 2008a; Whigham & Dick, 2009) and the effect of elitism on solution size, especially in later generations, is also shown in Figure 17. After 120 generations tree size starts to proliferate grossly, and it can be seen that this bloating is constrained proportionally to the elite fraction.

4.4 Quantifying Validity

The first generations tend to contain more invalid solutions, and once the first valid trajectories are found, an evolutionary process of refinement centred around these trajectories is triggered. In very hard to navigate scenarios, with densely packed barriers or narrow passages, the algorithm is unlikely to randomly land on a valid solution that can then be refined. Thus the rough-stage optimisation fails and the binary validation method is ineffective. Hildemann and Verstegen (2023) circumvents this problem by "seeding" vertices within a navigable channel, and then using GA and a repair mechanism to refine this rough progenitor trajectory. This requires some foreknowledge of the barrier layout and a more automated and potentially generalisable solution is for the algorithm to recognise such navigable channels by itself. Thus, in order to mould the trajectory into the allowed airspace of a highly complex environment, it is necessary to nudge the optimisation into the right direction by giving it more explicit feedback. Solutions that intersect the barrier geometry to different extents would then have to be penalised differently *i.e.* the validation criterion must be quantified.

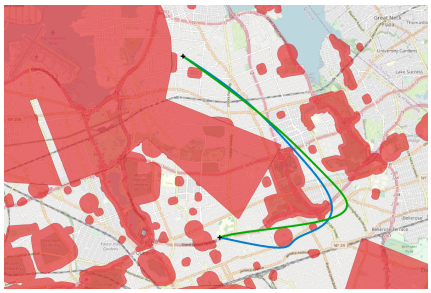


Figure 18: Solutions only obtainable with quantified intersection.

Towards this end, an additional parameter was introduced to "allow" intersection. Enabling the `adaptive_mode` parameter means that the fitness of invalid/intersecting trajectories is not identically calculated, but depends on the amount of intersection. Trajectories that intersect barriers over most of their length will thus be less fit than those that intersect only in a small part. By discriminating between solutions based on the size of the intersecting set, the refining evolutionary process can begin before a valid solution is found. This refinement is then driven by incremental improvements in fitness over incremental reductions in the size of the intersecting set. The algorithm is then able to find more intricate solution trajectories since it is able to know, from the size of the intersection, when it is evolving in the right direction.

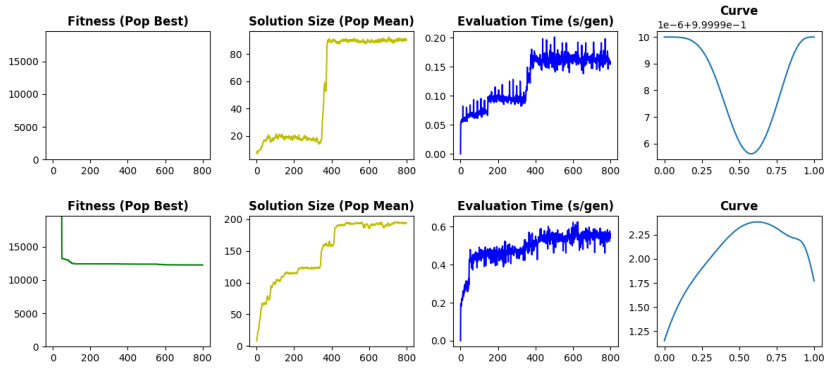


Figure 19: Optimisation with total invalidation (top) and flexible invalidation (bottom).

Cases have been observed where optimisation was unable to generate a valid solution due to the minimum required complexity of the solution. Figure 18 shows one example of how intersection quantification can "push" evolution towards a valid solution. This method is significantly slower because all intersections between the trajectory and barriers are measured and summed, instead of invalidating the trajectory after the first intersection is found. Figure 19 shows how quantifying the intersection facilitates the adaptation by encouraging individuals to adapt gradually to a niche. Since these solutions are not completely invalidated, it may happen that the final solution is invalid, having only minimised but not eliminated the amount of intersection. To avoid this, the `intersection_cost` parameter may be adjusted to impose a sufficient penalty to evolve away from even small intersections, towards a fully valid trajectory.

4.5 Geometry Complexity

The nature of the validation algorithm is such that every barrier is compared to every part of the solution geometry to confirm non-intersection, unless an intersection is encountered. Therefore, barrier geometries are processed continuously in memory and their size significantly affects the speed of validation. Simplifying barrier geometries can reduce the number of vertices drastically, with no appreciable or significant loss of information. Decimating geometries using the Douglas-Peucker algorithm can yield nearly functionally identical geometries with a much lighter memory requirement. Figure 20 shows a barrier polygon simplified with a maximum deviation of 10 m from the original, with the number of vertices reduced from 322 to 29. Total vertices in the entire set of polygons is reduced from 3218 to

284 (91%). Alternatively the geometries can be generalised given a threshold tolerance, to yield bounding geometries that are marginally larger but geometrically simpler. It can also be argued that this approach imposes an additional safety factor by slightly buffering the actual restricted zone. The result, however, was a mere 5% reduction in processing time for the simplified polygons.

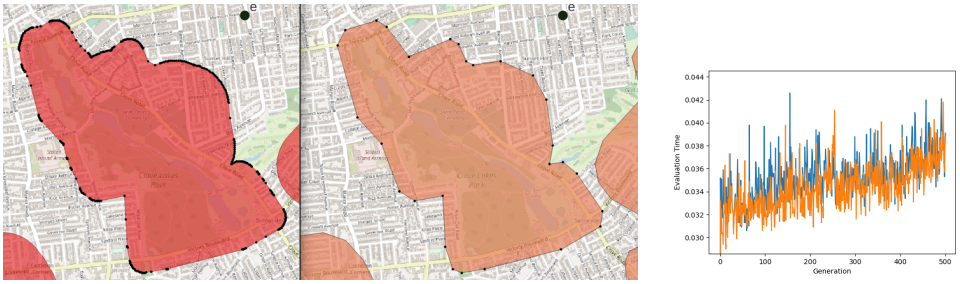


Figure 20: Vertex count comparison before (left) and after (right) geometry simplification. The graph shows the evaluation time for unsimplified (blue) and simplified (orange) geometries.

The optimal number of vertices to use depends on the scale of the area under study, since the vertex density along the trajectory depends directly on its length. Figure 21 shows the vertex density along a 9,000m route with approximately 9m per segment and 90m per segment, along with the optimisation duration in each case. Computation time scales proportionally with the vertex count and here a nearly 70% drop in speed is observed for a tenfold increase in vertex count.

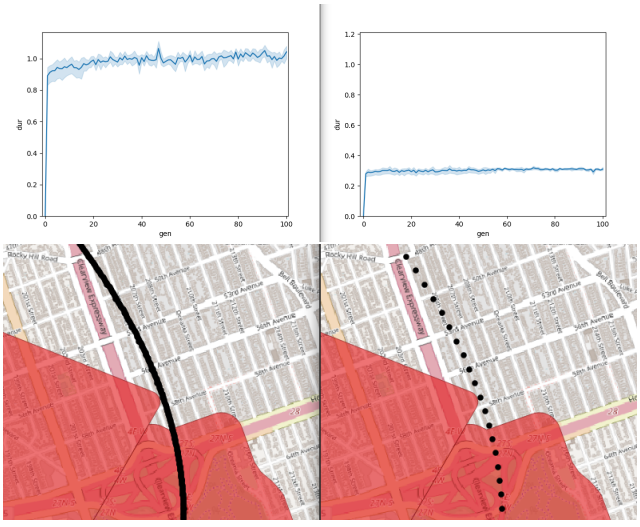


Figure 21: Trajectory waypoints with the number of segments between 1000 and 100.

4.6 Comparison of Validation Algorithms

Three validation methods for 3D intersections were developed, each with their respective strengths and weaknesses. They were compared by applying all three to the same simple trajectories within the Clove Lakes scenario. The Shapely-based 2.5D method is the fastest, ranging from 20 to 50 seconds for a 100-generation optimisation. The first SciPy-based method using Delaunay simplices was the slowest, requiring 2 to 5 hours for the same task. The second SciPy method using convex hulls required between 40 and 100 minutes.

5 Discussion

5.1 Function tree representation

The operators in the primitive set have been selected in order to maximise the variability achievable relative to the program size. Basic arithmetic functions addition, subtraction, multiplication, division and negation are used in conjunction with the sine and cosine trigonometric functions to produce solution curves that can theoretically match any trajectory shape. Other functions were tested, but ultimately excluded due to buffer overflows and performance degradation, and include roots, logarithms, exponents, powers and the hyperbolic tangent. Ephemeral constants consisting of floating point numbers were tested, but degraded performance. Integer ephemeral constants in a broader range can be used, but their effect on the optimisation has not been investigated.

In this work loosely typed GP is used, which allows any primitive or terminal to be passed as an argument to a primitive. Using strongly typed GP allows additional primitives to be incorporated that take or return, for example, Booleans. In such an implementation conditionals such as `if_then_else`, operators such as `xor` or Boolean terminals could also be used in the syntax tree. The efficacy of strongly typed GP applied to trajectory optimisation also remains to be investigated.

An alternative approach exists for creating genetic programs with two distinct outputs when given the same input *i.e.* varying a trajectory in two dimensions. In this work the primitive set is augmented with a second input argument, and alternatively passing a placeholder argument allows two separate values to be extracted. However, individuals can also be initialised as pairs of two function trees that evolve in tandem, with a single fitness value calculated for the pair. The crossover and mutation functions then need to be similarly modified to process the function pairs. This approach is more difficult to implement and tends to larger solutions, and its efficacy has not been investigated.

5.2 Solution validity

Ideally a population of valid trajectories would be initialised before the evolutionary process begins. Subsequent generations would then refine these individuals by adapting them more precisely to the problem. However, in highly obstructed routes it can be nearly impossible to randomly generate a valid geometry due to the complexity required. In these cases invalid solutions are kept and their fitness evaluated, with a penalty applied proportional to the "magnitude" of their invalidity. Doing so allows the evolution to find niches, or to incrementally solve a very complex optimisation that would not be possible without informative feedback. Another option is to seed waypoints within the navigable airspace by generating random points in the study area, and using these as checkpoints for the optimisation.

5.3 Calibration

Parameter influence was discernible to various extents, and it was possible to broadly calibrate the model to differing use cases by prioritising speed, accuracy or search radius. Certain parameters proved to be disproportionately significant, especially population size and parameters directly enhancing diversity.

Taking into account the results of testing different parameter values it is possible to identify configurations that are more or less optimally suited to the 3D trajectory optimisation problem. Population size should be as large as possible, preferably 1000 or more, and no less than 300. Size limits and variation probabilities depend on the complexity of the scenario, but may be safely increased to 17 for tree height, 90% for crossover and 25% for mutation. Elitism should be implemented with an elite fraction of 5-10%.

5.4 Applicability and scalability

The case studies presented vary in trajectory length from 5 to 25 km and represent a common application for low-altitude routing over moderately large distances in a highly obstructed space. In the areas studied, the universal height limit has been ignored in some cases in order to demonstrate the ability of the method to find trajectories over restricted areas.

It is possible to incorporate topographic models to incorporate the terrain as an additional restriction. In 2 dimensions the method can be applied to maritime routing, robot path planning or autonomous vehicles. In 3 dimensions the method can work for UAVs and submarines.

Scaling the method to much larger areas should have a very limited effect on speed, provided that the geometrical complexity is not excessive. Performance depends on the number of features and vertices within features, as well as the granularity of the trajectory and the desired coordinate precision. Geometries that are geographically larger or smaller will require the same amount of computational effort if the feature and vertex counts are similar.

5.5 Limitations and potential solutions

The tendency towards local optima can be effectively offset by larger population sizes and increased diversity. This translates directly into a slower optimisation but guarantees more consistent results despite the random nature of the initialisation and variation routines. The broadness of the search and the quality of the end result relies more on the starting parameters than on the length of the optimisation.

Very complicated trajectories can have an enormous computational cost, due to the tendency for bloating to occur and the limited feedback available to the selection algorithm. In these cases the optimisation is no more than a random search. Hildemann and Verstegen (2023) uses a seeding method to create a progenitor trajectory that is then refined incrementally using GA and repair. Although this method works well, the seeding requires prior knowledge of the environment. Quantifying intersection solves this problem in the scenarios tested, but requires significantly more solving time. One semi-automatic method of accommodating problems that require a very wide lateral or vertical deviation is to iteratively search for the nearest non-intersecting point in the plane that contains the midpoint of the trajectory start and end points and is normal to the line connecting them. Using the distance of this point and the displacement between the start and midpoint, a stretching factor can be calculated and used to exaggerate the route geometry in a certain direction. In this way the algorithm could yield trajectories with very wide turns early on, allowing the rest of the optimisation (refinement) to be triggered.

Where the length of intersection is measured in the evaluation routine (when adaptive mode is enabled) the algorithm is currently susceptible to major errors if overlapping polygons

are present. Parts of the trajectory that intersect multiple polygons will contribute to the total intersection length for each polygon. Care must be taken that no barrier geometries overlap if this setting is enabled.

One possible performance augmentation is spatial indexing of the barrier geometries. With this implemented, non-intersecting barriers can be identified without calling an intersection algorithm if a barrier is sufficiently distant. This can potentially speed up the validation process, which accounts for 85-95% of the processing time.

PostGIS is a popular geospatial extension for PostgreSQL that supports 3D geometries and 3D geometric predicates. Barriers can be represented as 3D polyhedral surfaces or triangulated irregular networks, and trajectories as 3D line segments (Real et al., 2019). It should be possible to recreate this research using geographic objects stored in a database and substituting the 3D intersection algorithm with the equivalent function in PostGIS.

It is not necessarily required for outputs to be converted to geographic lines before their intersection status can be established. With barriers modelled as sets of planes and paths as simple 3D linestrings, it is possible to run the intersection check in a purely geometric (mathematical) space. Comparison of this method with an explicitly geographical method will yield an indication of their respective merits. It is also possible to compute the geographic trajectory length without converting to a geographic format, by scaling the curve with the ratio of the size of the geographic interval to the size of the interval in the solution space.

Another promising prospect is meta-evolutionary algorithms, wherein a GA is used to optimise the parameters of a GP. The calibration done during this research could theoretically be done to a more exhaustive extent by an evolutionary algorithm operating over repeated runs of the same optimisation. Similar to this is co-evolutionary algorithms that take into account multiple UAVs, and have the added condition of non-collision with other UAVs while travelling the optimised route.

The cost function is able to incorporate four objectives with varying weights, although overall trajectory length is the dominant fitness criterion. For improved multi-objective optimisation, it is necessary to alter the method of selection in order to obtain a Pareto front. NSGA-III is a novel non-dominated sorting approach, based on NSGA-II (used successfully by Hildemann and Verstegen (2023) for 3D routing) that has shown its effectiveness with up to 15 objectives (Deb & Jain, 2013), and presents a promising research opportunity for 3D trajectory optimisation in the future.

6 Conclusion

In this work, a novel approach to the trajectory optimisation problem in 3D was proposed, using genetic programs to encode geometries and applying selective pressures to minimise the cost of the route while avoiding intersection with obstacles. The method was applied to two case studies in New York with widely differing levels of navigational difficulty, and with different constraints imposed to assess the adaptability of the algorithm.

The literature suggests that trajectory optimisation is a well-studied topic and that various methods using artificial intelligence are in current employment, including evolutionary algorithms. However, it is also very clear that most current methods are limited to planar environments, discontinuous spaces, or both. Furthermore, the studies addressing the problem in three dimensions and continuous space have been limited in terms of speed and automaticity. A promising research opportunity therefore exists in applying genetic programming to the trajectory optimisation problem.

After devising a method of evolving functions and transforming them into geographic trajectories, it was possible to evolve geographic trajectories in place and subsequently answer the research questions. The syntax-tree representation of functions encoding curves proved to be able to represent highly variable shapes to match complicated environments rich in obstacles. These trees are also easily processed and stored, and amenable to manipulation by evolutionary processes. Compared to Hildemann and Verstegen (2023) the optimisation improved upon the computational requirement, and allowed results to be obtained in a much shorter time. The optimisation process was successful in every use case presented, and showed a reliable convergence over numerous experiments. The solutions are also lightweight and the bloat-control methods implemented were able to minimise their size and maximise their processing speed.

In complex use cases it was necessary to guide the evolution towards validity, as in the presence of too many barriers the first-generation initialisation of a valid geometry becomes exceedingly unlikely. An adaptive method, which incorporates the length of intersection into the cost function, was able to find heuristically optimal trajectories in very confined spaces. An alternative to this may be to construct a seed function that operates in the initialisation process, to increase the diversity in first generation individual creation.

Three methods of 3D geometry intersection were compared, with a clear inverse relationship between processing speed and geometry complexity. The shapely-based intersection evaluation method is faster, but restricts the resolvable detail of the third dimension. A significant performance cost accompanied the fully 3D methods, and as a result these may be impractical in some use cases. The 3D representation of barrier geometries is limited to convex shapes, although this only means that additional preprocessing is required to accommodate concave polyhedrons. Full mesh geometry support is an opportunity for future work, as is the investigation into the possible benefit of GPU processing.

This research not only showcases the versatility of GP, but also demonstrates the possibility of solving spatial optimisation problems where the form of the solution being found differs widely from the final form. The only requirement is that the solution, or an encoded form of the solution, is derivable from a function.

References

Ait Saadi, A., Soukane, A., Meraihi, Y., Benmessaoud Gabis, A., Mirjalili, S., & Ramdane-Cherif, A. (2022). Uav path planning using optimization approaches: A survey. *Archives of Computational Methods in Engineering*, 1–52.

Alves, N., Ferreira, M. A. S., Colombini, E. L., Da Silva Simoes, A., et al. (2020). An evolutionary algorithm for quadcopter trajectory optimization in aerial challenges. *2020 Latin American Robotics Symposium, 2020 Brazilian Symposium on Robotics and 2020 Workshop on Robotics in Education, LARS-SBR-WRE 2020*.

An, P. (2018). Path optimization method of autonomous intelligent obstacle avoidance for multi-joint submarine robot. *Journal of Coastal Research*, (82 (10082)), 288–293.

Baker, B. M., & Ayechew, M. (2003). A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, *30*(5), 787–800.

Balicki, J. (2006). Multicriterion genetic programming for trajectory planning of underwater vehicle. *IJCSNS*, *6*(12), 1.

Balogun, A.-L., Matori, A.-N., Hamid-Mosaku, A. I., Umar Lawal, D., & Ahmed Chandio, I. (2017). Fuzzy mcdm-based gis model for subsea oil pipeline route optimization: An integrated approach. *Marine Georesources & Geotechnology*, *35*(7), 961–969.

Behzadi, S., & Alesheikh, A. A. (2008). A pseudo genetic algorithm for solving best path problem. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *3*.

Cakir, M. (2015). 2d path planning of uavs with genetic algorithm in a constrained environment. *2015 6th International Conference on Modeling, Simulation, and Applied Optimization (ICMSAO)*, 1–5.

Deb, K., & Jain, H. (2013). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE transactions on evolutionary computation*, *18*(4), 577–601.

Edison, E., & Shima, T. (2011). Integrated task assignment and path optimization for cooperating uninhabited aerial vehicles using genetic algorithms. *Computers & Operations Research*, *38*(1), 340–356.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., & Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, *13*, 2171–2175.

García, A. E., González, H. E., & Schupke, D. (2022). Hybrid route optimisation for maximum air to ground channel quality [31]. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *105*(2).

Garg, M., Kumar, A., & Sujit, P. (2015). Terrain-based landing site selection and path planning for fixed-wing uavs. *2015 international conference on unmanned aircraft systems (ICUAS)*, 246–251.

Garip, Z., Karayel, D., & Erhan Çimen, M. (2022). A study on path planning optimization of mobile robots based on hybrid algorithm. *Concurrency and Computation: Practice and Experience*, *34*(5), e6721.

Hanshar, F. T., & Ombuki-Berman, B. M. (2007). Dynamic vehicle routing using genetic algorithms. *Applied Intelligence*, *27*(1), 89–99.

Hildemann, M., & Verstegen, J. A. (2023). 3d-flight route optimization for air-taxis in urban areas with evolutionary algorithms and gis. *Journal of Air Transport Management*, *107*, 102356.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press.

Hu, X.-B., Wu, S.-F., & Jiang, J. (2004). On-line free-flight path optimization based on improved genetic algorithms. *Engineering Applications of Artificial Intelligence*, *17*(8), 897–907.

Huynh, D. Q. (2009). Metrics for 3d rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, *35*, 155–164.

Jing, Y., Luo, C., & Liu, G. (2022). Multiobjective path optimization for autonomous land levelling operations based on an improved moea/d-aco [106995]. *Computers and Electronics in Agriculture*, *197*.

Kala, R. (2012). Multi-robot path planning using co-evolutionary genetic programming. *Expert Systems with Applications*, *39*(3), 3817–3831.

Kim, H., Kim, S.-H., Jeon, M., Kim, J., Song, S., & Paik, K.-J. (2017). A study on path optimization method of an unmanned surface vehicle under environmental loads using genetic algorithm. *Ocean Engineering*, *142*, 616–624.

Koza, J. (1992). On the programming of computers by means of natural selection. *Genetic programming*.

Kumar, A., et al. (2014). Efficient hierarchical hybrids parallel genetic algorithm for shortest path routing. *2014 5th International Conference-Confluence The Next Generation Information Technology Summit (Confluence)*, 257–261.

Luke, S., & Panait, L. (2002). Fighting bloat with nonparametric parsimony pressure. *International conference on parallel problem solving from nature*, 411–421.

Ma, J., Liu, Y., Zang, S., & Wang, L. (2020). Robot path planning based on genetic algorithm fused with continuous bezier optimization. *Computational intelligence and neuroscience*, *2020*.

Mane, S. B., & Vhanale, S. (2019). Genetic algorithm approach for obstacle avoidance and path optimization of mobile robot. In *Computing, communication and signal processing* (pp. 649–659). Springer.

Mansur, V., Reddy, S., Sujatha, R., & Sujatha, R. (2020). Deploying complementary filter to avert gimbal lock in drones using quaternion angles. *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*, 751–756.

Meng, H., & Xin, G. (2010). Uav route planning based on the genetic simulated annealing algorithm. *2010 IEEE International Conference on Mechatronics and Automation*, 788–793.

Mokhtari, S. A. (2022). Fopid control of quadrotor based on neural networks optimization and path planning through machine learning and pso algorithm. *International Journal of Aeronautical and Space Sciences*, 1–16.

Oultiligh, A., Ayad, H., Elkari, A., & Mjahed, M. (2020). Path planning using particle swarm optimization and fuzzy logic. *International Conference on Artificial Intelligence & Industrial Applications*, 239–251.

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., et al. (2021). The prisma 2020 statement: An updated guideline for reporting systematic reviews. *Systematic reviews*, *10*(1), 1–11.

Pérez-Cutiño, M. A., Rodríguez, F., Pascual, L. D., & Díaz-Báñez, J. M. (2022). Ornithopter trajectory optimization with neural networks and random forest [17]. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *105*(1).

Pezer, D. (2016). Efficiency of tool path optimization using genetic algorithm in relation to the optimization achieved with the cam software. *Procedia Engineering*, *149*, 374–379.

Poli, R., Langdon, W. B., McPhee, N. F., & Koza, J. R. (2008). A field guide to genetic programming.

Poli, R., McPhee, N. F., & Vanneschi, L. (2008a). Analysis of theeffects ofelitismonbloat inlinear and tree-basedgenetic programming. In *Genetic programming theory and practice vi* (pp. 1–20). Springer.

Poli, R., McPhee, N. F., & Vanneschi, L. (2008b). Elitism reduces bloat in genetic programming. *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, 1343–1344.

Qie, H., Shi, D., Shen, T., Xu, X., Li, Y., & Wang, L. (2019). Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning. *IEEE access*, *7*, 146264–146272.

Rath, A. K., Parhi, D. R., Das, H. C., Kumar, P. B., Muni, M. K., & Salony, K. (2019). Path optimization for navigation of a humanoid robot using hybridized fuzzy-genetic algorithm. *International journal of intelligent unmanned systems*, *7*(3), 112–119.

Real, L. C. V., Silva, B., Meliksetian, D. S., & Sacchi, K. (2019). Large-scale 3d geospatial processing made possible. *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 199–208.

Sandurkar, S., & Chen, W. (1999). Gaprus—genetic algorithms based pipe routing using tessellated objects. *Computers in industry*, *38*(3), 209–223.

Sivanandam, S., & Deepa, S. (2008). Genetic algorithms. In *Introduction to genetic algorithms* (pp. 15–37). Springer.

Sundaran, K. (2018). Genetic algorithm based optimization technique for route planning of wheeled mobile robot. *2018 Fourth International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, 1–5.

Villarrubia, G., De Paz, J. F., Chamoso, P., & De la Prieta, F. (2018). Artificial neural networks used in optimization problems. *Neurocomputing*, *272*, 10–16.

Wang, H., Lyu, W., Yao, P., Liang, X., & Liu, C. (2015). Three-dimensional path planning for unmanned aerial vehicle based on interfered fluid dynamical system. *Chinese Journal of Aeronautics*, *28*(1), 229–239.

Whigham, P. A., & Dick, G. (2009). Implicitly controlling bloat in genetic programming. *IEEE Transactions on Evolutionary Computation*, *14*(2), 173–190.

Yang, X., Cai, M., & Li, J. (2016). Path planning for unmanned aerial vehicles based on genetic programming. *2016 Chinese Control and Decision Conference (CCDC)*, 717–722.

Zhang, B., & Duan, H. (2015). Three-dimensional path planning for uninhabited combat aerial vehicle based on predator-prey pigeon-inspired optimization in dynamic environment. *IEEE/ACM transactions on computational biology and bioinformatics*, *14*(1), 97–107.