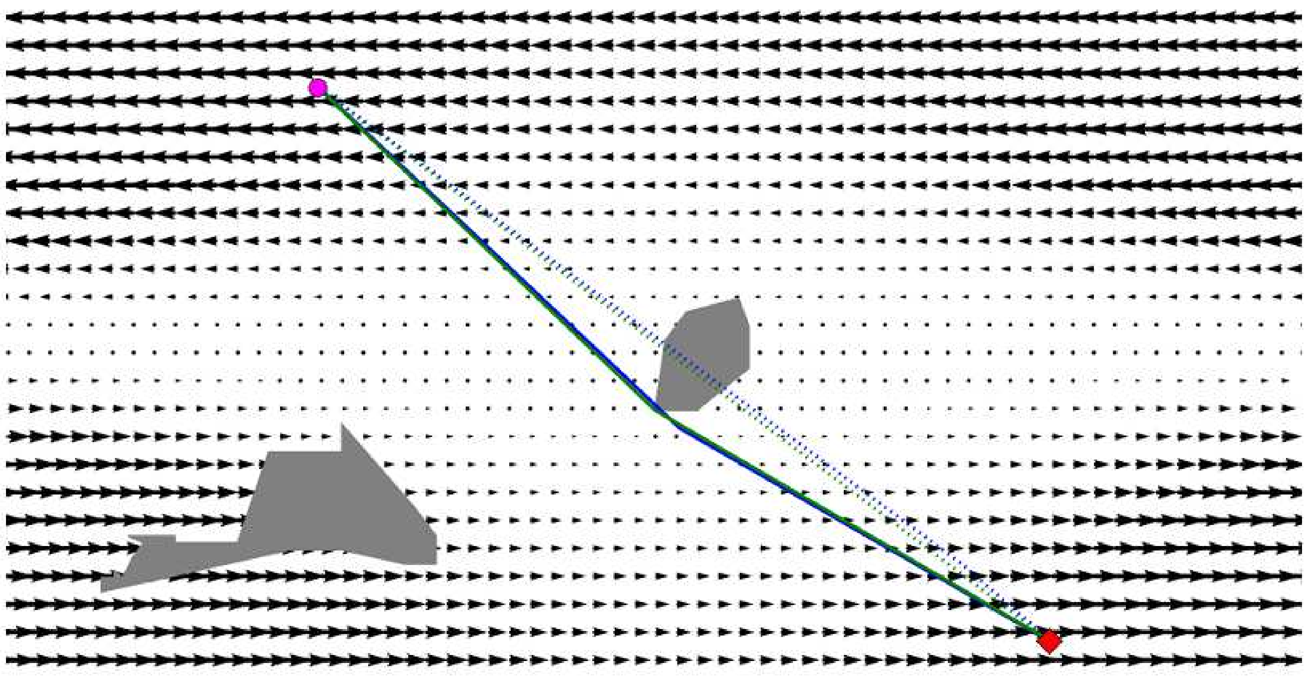
Faculty of Aerospace Engineering

Four-Dimensional Trajectory Planning in Air Traffic Management

Feasibility of a Heuristic Branching Method

D. F. Rein-Weston

January 12, 2017



Delft University of Technology

Challenge the future

Four-Dimensional Trajectory Planning in Air Traffic Management

Feasibility of a Heuristic Branching Method

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

D. F. Rein-Weston

January 12, 2017

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

Delft University Of Technology
Department Of
Control and Operations

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance a thesis entitled **"Four-Dimensional Trajectory Planning in Air Traffic Management"** by **D. F. Rein-Weston** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: January 12, 2017

Readers:

Prof. dr. ir. J. M. Hoekstra

Dr. ir. J. Ellerbroek

Preface

This work represents the culmination of my master's degree studies in Aerospace Engineering at TU Delft. The experience of studying here has been an incredibly rewarding one, and I am grateful to several individuals for their involvement.

First, I would like to thank my thesis advisors, Prof. dr. ir. Jacco Hoekstra and Dr. ir. Joost Ellerbroek, for their guidance during this project. It was an honor to work with both of them. I am grateful to my advisors for showing trust in my work, while at the same time offering challenging insights to push my work forward; I will remember with fondness the often animated discussions held between my advisors in front of the whiteboard in Professor Hoekstra's office. Second, I would also like to thank the many teachers, not only within Control & Operations but also outside the department, from whom I learned during my coursework at TU Delft. Furthermore, I would like to thank Bertine Markus for her logistical help throughout.

I had the privilege of working with talented classmates, and wish to thank them for everything they taught me. In particular, however, thank you to Apeksha and Pulkit for being not only my study companions but also close friends.

I am very grateful to my mother and father for their endless support, and for making the opportunity to study in Delft possible. I would also like to thank my sisters, Annie and Claire, both of whom I admire very much. Finally, I would like to thank Chris for his constant encouragement, and for joining me on this adventure abroad.

Daphne Rein-Weston
Delft University of Technology
January 12, 2017

Contents

Acronyms

4DT	Four-Dimensional Trajectory
4PL	Four Parameter Logistic Curve
ADS-B	Automatic Dependent Surveillance-Broadcast
AOC	Airline Operations Center
AOP	Autonomous Operations Planner
ATFM	Air Traffic Flow Management
ATM	Air Traffic Management
CPA	Closest Point of Approach
CTA	Controlled Time of Arrival
DUT	Delft University of Technology
FAA	Federal Aviation Administration
FMS	Flight Management System
I4D	Initial 4D Project
ICAO	International Civil Aviation Organization
NextGen	Next Generation Air Transportation System
PSO	Particle Swarm Optimization
RBT	Reference Business Trajectory
RTA	Required Time of Arrival
SA	Simulated Annealing
SESAR	Single European Sky ATM Research
TASAR	Traffic Aware Strategic Aircrew Requests
TBO	Trajectory-Based Operations
TEMO	Time and Energy Managed Operations
UAV	Unmanned Aerial Vehicle

List of Figures

Part I

Technical Paper

Four-Dimensional Trajectory Planning in Air Traffic Management: Feasibility of a Heuristic Branching Method

Daphne Rein-Weston,[*] Joost Ellerbroek[†] and Jacco Hoekstra[‡]

Delft University of Technology, Delft, South Holland, 2629 HS, The Netherlands

A shift from the current Air Traffic Management (ATM) operational scheme to Trajectory-Based Operations (TBO) is imminent in both the United States and in Europe. An enabling technology is the four-dimensional trajectory (4DT) planner, which is the focus of this research. The aim is to consider the feasibility of a heuristic branching method for 4DT planning by developing such a planner and testing it against an established Dijkstra-like approach. Windfield and obstacle field complexities are varied and the solution effectiveness and runtimes of the tested planners are compared by the developed metrics. No significant statistical difference is found between the distributions of results for solution effectiveness of the two planners, but a significant reduction in runtime is shown by the newly developed planner. The study thus indicates the feasibility of a heuristic branching-based method for 4DT planning in the realm of ATM, which informs future research in this domain.

I. Introduction

The air transportation systems of the United States and Europe are facing significant challenges under increased demand. The modernization of Air Traffic Management (ATM) systems is vital to the safe accommodation of this demand.[1] Through their respective initiatives, namely the Next Generation Air Transportation System (NextGen) and the Single European Sky ATM Research Program (SESAR), the United States and Europe are working toward several common goals. In specific, the 2014 NextGen-SESAR State of Harmonisation Document states that "The goals on each side of the Atlantic are to improve overall aviation system performance, particularly in the areas of flight efficiency and environmental impact, while also meeting expected demands for increased capacity and continuing to maintain the highest levels of safety" [1, pp. 8]. A shift from present-day operations to Trajectory-Based Operations (TBO) is envisioned by both NextGen and SESAR to meet the aforementioned goals. ATM under TBO will no longer suffer from inefficiencies of generic predefined routes, but instead will cater toward capabilities of individual aircraft. Not only is efficiency addressed in this way, but also environmental impact: for example, continuous climbs and descents (enabled by TBO) offer significant reductions in fuel and emission.[2] This performance-based operational scheme is also expected to offer an increase in capacity due to a reduction in trajectory uncertainty.[3] According to the International Civil Aviation Organization (ICAO), TBO is slated for approval in 2028.[4]

The building blocks of TBO are four-dimensional trajectories (4DTs): a 4DT is a trajectory defined in 3D-space (latitude, longitude, altitude) as well as time. The 4DT provides the basis for separation management decisions in TBO.[3] Since intent is clearly stipulated in the trajectory definition and the trajectory definition is shared with all ATM stakeholders, the imminent shift toward TBO will result in a more proactive ATM system. For example, instead of reacting to traffic or other obstacles that arise as an aircraft is following its flight plan, a trajectory under TBO takes into account anticipated conflicts and therefore shifts the operations into a more strategic rather than tactical realm. This future ATM concept thus relies on the ability to plan aircraft trajectories such that **a)** the efficiency of the individual flight is maximized while

[*]MSc Student, Aerospace Engineering: Control and Simulation.
[†]Assistant Professor, Aerospace Engineering: Control and Simulation.
[‡]Professor, Aerospace Engineering: Control and Simulation.

b) safety is ensured by avoiding interactions between trajectories and other trajectories, as well as between trajectories and other potential hazards. A trajectory planner, therefore, is an enabling technology that is crucial to the viability of TBO in real life operations.

Several established path planning methods with relevance to the previously described context will be reviewed next, in Section A. More details regarding the aims of this particular research will be provided in Section B.

A. Theoretical Background

There are several ingredients inherent to planning that are involved in most planning algorithms. According to S. M. LaValle, these ingredients include: state, time, actions, initial & goal states, a criterion, and a plan.[5] In the brief overview of planning methods offered in Section 1, it can be assumed that the state is a position, actions are the movement from one state to the next, and the initial and goal states are referred to as the origin and destination, respectively. With regard to the other ingredients, time is not explicitly modeled (but is implied by the sequencing of actions), the criterion is distance optimality, and the resulting plan is an optimal path from the origin to the destination.

Sections 2 and 3 detail the applications of path planning algorithms to the context of 4DT planning in ATM, as applied by B. Girardet[6] and by the current authors, respectively.

1. Survey of Planning Methods

The strategies for finding the optimal path from origin to destination can be divided into three main categories: graph-search methods, gradient-based methods, and stochastic search methods. A brief introduction to methods of the first two categories will be given here, while an example of a stochastic search method will be given in the next section.

As indicated by its name, graph-search methods use a graph to solve the shortest path problem. A graph is defined as ordered pairs of vertices and edges, while a path is "an ordered sequence of vertices, such that an edge exists between two successive vertices" [7, pp. 1]. The shortest path from one particular vertex to another is the path between those vertices with the least associated cost. Algorithms such as Dijkstra's algorithm are designed to solve for the shortest path by computing the minimum path weight. Dijkstra's algorithm solves the shortest path problem with a time complexity of $\mathcal{O}(N^2)$.[7] As the number of vertices in the graph increases, Dijkstra's algorithm becomes computationally expensive at a drastic rate and thus is limited in terms of operational viability. One way of addressing this limitation is by directing the search rather than considering all nodes in an uninformed manner. The AStar algorithm is an example of such an approach, where the use of a well-chosen heuristic guides the search. A benefit of the graph-search methods is that a global solution will be found; a potential detriment is the inherent need for a discretization of the problem space.

Another category of search methods are gradient-based methods, such as potential fields.[8] Such methods are characterized by a local, rather than global search. The combination of a gradient-based method with a more global driving algorithm can yield a powerful planner. It can be shown that this is the approach taken by B. Girardet with the combination of an optimal control formulation of the problem with a Dijkstra-like algorithm, as introduced in the next section.

2. Ordered Upwind Method: Application by B. Girardet

The work of B. Girardet, particularly the doctoral dissertation of 2014, is the inspiration for this study, and thus is a key reference to review.[6] To introduce the algorithm that is central to Girardet's dissertation, an earlier Girardet publication is referenced here.[9]

The Ordered Upwind algorithm, originally developed by Sethian and Vladimirsky, is described by B. Girardet as "avoid[ing] iterations through a careful use of the information about the characteristic directions of the PDE" [9, pp. 5]. At the basis of this method is a Hamilton-Jacobi equation that can be formulated as a "front propagation problem" [10, pp. 336]. Although the Ordered Upwind method involves steps that are similar to Dijkstra's algorithm, it "converges towards the exact solution of the Hamilton-Jacobi equation" [9, pp. 6]. The complexity of the Ordered Upwind method, which depends on the number of mesh points N, is $\mathcal{O}(N \log N)$.[9] The Ordered Upwind algorithm offers a clever alternative to otherwise costly numerical methods. In specific, the Ordered Upwind algorithm can be considered as an adaptation of the Fast

Marching Method, whose basic principle is highlighted in Figure 1. Instead of tracking a moving interface (depicted on the left in Figure 1), the interface is treated as static and nodes are systematically (in a similar way to Dijkstra's algorithm) updated with "crossing time" information.[11]

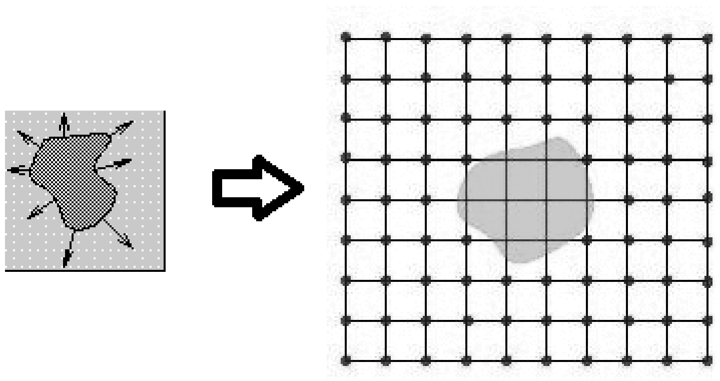


Figure 1: Illustrations from Sethian's description of Fast Marching methods[11]

In the case of the wind-optimal trajectory problem, however, the speed of the aircraft depends on direction; therefore, the speed is not considered isotropic and the Fast Marching methods are not applicable. As previously mentioned, the Ordered Upwind method is based on the same principles of the Fast Marching methods, but has been adapted to take into account the "anisotropy ratio" (i.e. the ratio of fastest and slowest speed at a node) to compute the next unknown node information.[11] It is recommended to refer to the original work of B. Girardet for a better understanding of the ordered, Dijkstra-like movement between nodes through the provided description of the steps in calculating the value function.[9]

The overall approach taken by Girardet in the 2014 dissertation is to first optimize a single trajectory using the Ordered Upwind method and then to consider all trajectories in order to make adjustments for de-congestion.[6] For the de-congestion step, Girardet makes use of simulated annealing, which is a stochastic search method. A main reason for implementing the stochastic search method is the problem size; for example, there can be 500 trajectories to optimize on a given flight level [6, pp. 130]. Stochastic search methods, such as simulated annealing, have the great benefit of a computational time that is independent of the problem complexity.[12] Nonetheless, it is suggested by Girardet as future work that computation time could be improved.[6] In addition to a lengthy computation time, simulated annealing in an ATM application may suffer from a lack of repeatability; the element of randomness in the algorithm may hinder, for example, the coordination between ATM stakeholders that is expected by ICAO.[13]

3. Heuristic Branching Method

As mentioned in Section 1, the undirected searches similar to Dijkstra's algorithm suffer in terms of computation time. However, an alternative solution is to "consider using some kind of enumeration procedure [that is] cleverly structured so that only a tiny fraction of the feasible solutions actually need be examined," such as the branch-and-bound method [14, pp. 501]. The branching portion of the branch-and-bound method is to subdivide the feasible solutions and the bounding portion is to define for each subset "how good its best feasible solution can be" [14, pp. 503]. The branch-and-bound method enables a "shortest path" to therefore be found without iterating through all combinations of nodes. The order in which to process sub-problems created by branching is defined by the chosen search strategy; two examples of search strategies are the best-first search and the depth-first search.[15] In the case of the best-first search, the number of sub-problems to consider is minimized, thus the method is relatively quick in improving the lower bound [15, pp. 269].

The work of A. Eele and A. Richards[16] in the area of path planning using a pragmatic branch-and-bound method has inspired the alternative planner that is central to this study and will be referred to in future sections.

B. Research Aim

The aim of this research is to consider the feasibility of a heuristic branching method for 4DT planning. In order to achieve this objective, a heuristic branching planner algorithm is developed and its performance, in particular computation speed and path effectiveness, is compared against that of the previously described Girardet method. It is hypothesized that given varying wind and obstacle fields, the heuristic branching planner will generate trajectories within tolerance of the Girardet method (i.e. complete trajectory generation and no statistically significant difference in path deviation measures) but with significant reduction in computation time and with predictable, transparent behavior. Therefore, the acceptance of the hypothesis would indicate the feasibility of the heuristic branching method for use in 4DT planning. An acceptance of the hypothesis would also indicate that further research in this area is merited.

The research method is detailed in Section II, followed by the results in Section III. A discussion section is given as Section IV, and precedes the conclusions given in Section V.

II. Method

As previously stated, the aim of this research is to consider the feasibility of a heuristic branching method for 4DT planning. Therefore, such a planner must be developed in order to be evaluated. The planner development is the crux of this research project; further details are provided in Section A.

In terms of evaluating the developed planner, a comparison test is performed between the heuristic branching planner and an implementation of the Girardet-method-based planner. Further details regarding the benchmark planner and the experiment variables are given in Section B.

A. Development of Planner using Heuristic Branching Method

The development of the heuristic branching method is built upon assumptions regarding available data. Namely, the origin and destination of the aircraft for which the planner is being invoked are assumed as givens. Additionally, the definitions of the obstacles around which the aircraft must navigate are known. It is further assumed that the definition of any obstacle to be avoided will be in the form of a set of vertices; this assumption is considered to be reasonable since "any arbitrary shaped obstacle can be modeled as [a] convex polygon" [17, pp. 7]. Thus, it is assumed that the input to the planner will be an obstacle field consisting of sets of vertices that define polygons along with the coordinates of the aircraft's desired origin and destination. The generally accepted definition of a 4DT would imply that any coordinate along the trajectory consists of a latitude, longitude, altitude, and time. However, for the purposes of this research, any coordinate being defined along the aircraft's planned trajectory will only consist of a latitude, longitude, and altitude. Time is implied by the sequencing of the waypoints along the route, coupled with the aircraft's ground speed. It should be noted that the aircraft's ground speed can be calculated from the true airspeed (assumed constant) and the given windfield. The windfield is a parameter that will be defined as a further input to the planner. A final simplification is made for the alignment of this planner with the benchmark planner; namely, a constant altitude will be assumed for the duration of the planned trajectory.

To illustrate the logic of the heuristic branching method developed for this research, Figures 2a to 2f are given below. The figures illustrate an example with three obstacles: obstacle A, B, and C. The criterion for the path planning problem in this example is to optimize for distance, or in other words, to minimize the distance traveled. Therefore, wind will not be considered in this demonstration of the logic behind the heuristic branching method. The adjustment needed to account for wind will be discussed at the end of this section.

The first step, depicted in Figure 2a, is to make a direct path between the origin and destination, without concern for any obstacle. This direct path is the initial trial solution. If no obstacles were to obstruct this path, it would also be the final solution and no further branching would be required. However, since obstacle A does interfere with the path shown in Figure 2a, a deviation about obstacle A is necessary and therefore obstacle A is referred to as the branching obstacle. The algorithm creates trial solutions based on clockwise (Figure 2b) and counterclockwise (Figure 2c) rotations about the branching obstacle. Before continuing, it should be mentioned that in the case of more than one obstacle obstructing a trial solution, it is the first encountered obstacle which becomes the branching obstacle. This decision has been made based on the findings of A. Eele and A. Richards.[16]

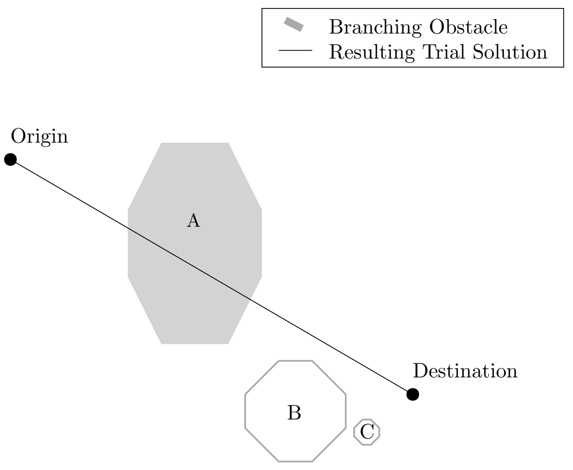


Figure 2a: Branching Algorithm Logic: Initial Trial Solution

Figure 2b shows the trial solution based on a clockwise (CW) rotation about the branching obstacle, obstacle A. Several steps are taken in order to arrive at this trial solution. First, the intersection points of the original direct route with obstacle A are taken as the starting and ending points of the CW rotation about obstacle A. In other words, the first intersection point of the original route with obstacle A is the "entry point" and the final intersection of the original route with obstacle A is the "exit point". The segments from the origin to the entry point and from the exit point to the destination are preserved, but an additional portion of the path is added to traverse the branching obstacle in a CW direction between the entry point and the exit point. The second step in creating the trial solution is to "snap" these entry and exit points to the nearest traversed obstacle node, as shown by the dotted route in Figure 2b. Before declaring this as a trial solution, a check is made to see if any unnecessary path segments can be discarded. This step is called the "backward cleanup," since it involves looking back to the already created path to see if any of the pivotal points, or waypoints, can be eliminated. The criterion for eliminating a waypoint in the backward cleanup is that the resulting path must not intersect any obstacle. It can be seen in Figure 2b that the new path segment created by discarding the first obstacle A node involved in the CW rotation does not intersect with any obstacles and is therefore accepted as part of the resulting trial solution. To eliminate the second CW rotation obstacle A node would, however, cause the path to intersect obstacle A and is therefore not acceptable. The backward cleanup step therefore retains nodes two and three, but discards nodes four and five. The solid path shown in Figure 2b represents the resulting trial solution. Although this trial solution would be a feasible solution for the entire problem, the algorithm requires a counterclockwise (CCW) rotation about the current branching obstacle to be investigated as well.

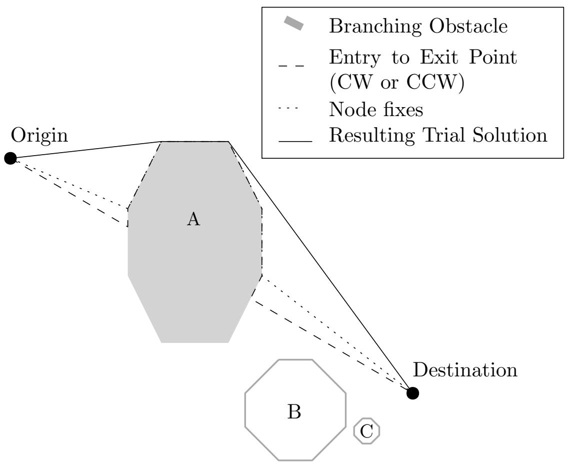


Figure 2b: Branching Algorithm Logic: Clockwise (CW) Obstacle A Avoidance

Figure 2c shows the the same steps just described, but with a CCW rotation about obstacle A. First, the original route is adapted to traverse obstacle A in a CCW direction between the entry and exit points. Second, the path is adjusted to traverse the obstacle not from these entry and exit points, but rather from the closest obstacle node already included in the adapted path. It should be noted that at this stage, it becomes clear that the resulting trial solution will intersect obstacle B. Third, the backward cleanup step minimizes path length by eliminating an unnecessary waypoint from the path.

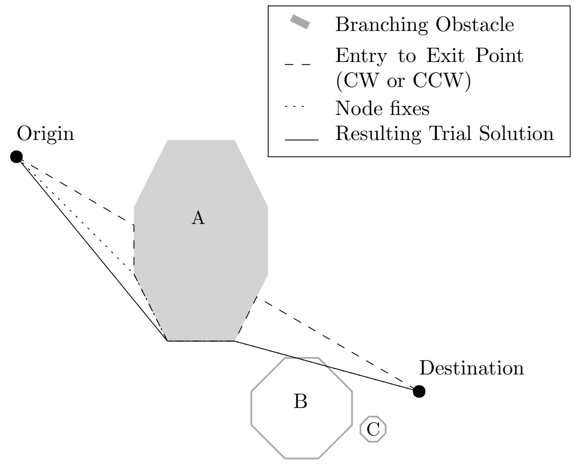


Figure 2c: Branching Algorithm Logic: Counterclockwise (CCW) Obstacle A Avoidance

The currently available trial solutions, namely the trial solutions shown in Figures 2b and 2c are each compared to the original path (shown in Figure 2a) from which they were created. The trial solution with the least additional path length is deemed "most promising" and is therefore investigated first. It should be noted that the decision to design the algorithm such that the path with least deviation is investigated first is inspired by the results of A. Eele and A. Richards.[16]

In Figure 2d, the process detailed from Figures 2a through 2c is begun again but with the trial solution of Figure 2c as the original path. Thus, Figure 2d shows the resulting trial solution based on a CW rotation about the branching obstacle B. Figure 2e shows the resulting trial solution based on a CCW rotation about the branching obstacle B.

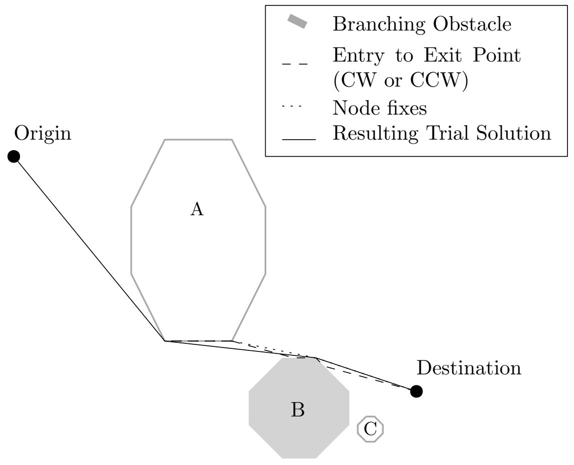


Figure 2d: Branching Algorithm Logic: Clockwise (CW) Obstacle B Avoidance

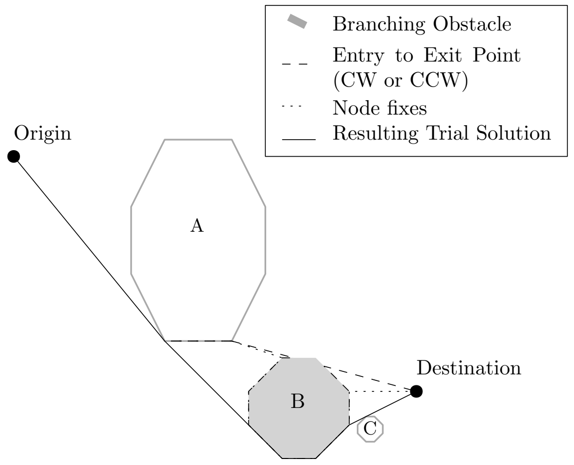


Figure 2e: Branching Algorithm Logic: Counterclockwise (CCW) Obstacle B Avoidance

It is now decided to further pursue the trial solution of Figure 2d because the extra path length from its original path (the trial solution of Figure 2c) is small compared to the extra path lengths of the other available trial solutions (Figures 2e and 2b) with respect to their respective original paths (Figures 2c and 2a). Therefore, the trial solution of Figure 2d is considered; this trial solution is already a feasible path (no further branching required) because it does not encounter any obstacles. The trial solution of Figure 2d therefore becomes the "incumbent" solution, meaning that its path length will be held as a standard against which any trial solutions considered in the future will be compared. In this example, the trial solution of Figure 2d becomes the final solution since the remaining trial solutions (Figures 2e and 2b) have longer path lengths and are therefore discarded. The final solution is shown in Figure 2f.

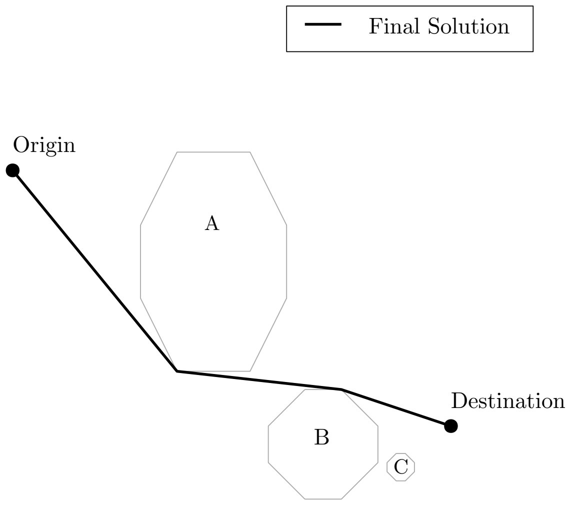


Figure 2f: Branching Algorithm Logic: Final Solution

The algorithm is further described through pseudocode in Algorithm 1 below. It should be noted that the "subproblem list" is a list of trial solutions that have yet to be investigated or discarded. Thus, the algorithm continues as long as the subproblem list contains at least one trial solution. As is commented in Algorithm 1, the selection of a trial solution removes it from the list while the creation of clockwise and counterclockwise alternatives adds trial solutions to the subproblem list.

Algorithm 1 Heuristic Branching Planner Pseudocode

```
Initialize subproblem list with direct route
while subproblem list is not empty
    select least deviation OR only option                          ▷ removes from subproblem list
    compare to incumbent (if incumbent exists)
    if intersects obstacle
        branch on first encountered obstacle
        clockwise and counterclockwise, direct to destination      ▷ adds to subproblem list
        backwards cleanup
        compare route length of each alternative to "parent" ($\delta$)
    else
        update incumbent
return incumbent
```

The combination of the intricacies involved in the creation of each trial solution, as detailed previously in Figures 2a to 2f, and the overall algorithm structure, as shown in Algorithm 1, form the branching-method planner. Prior to the evaluation phase, however, an additional option to optimize for time must be included. A main interest of the authors is to understand how the performance of the planner compares to that of the benchmark planner in the presence of wind; therefore, the optimization preference is for flight time rather than distance. To make this adjustment, the selection of a subproblem is based on least additional flight time rather than least extra distance. In other words, the comparison of a trial solution to its "parent", as described in Algorithm 1 results in a δ with units of time rather than distance. The flight time for a given segment is calculated by the known true airspeed (assumed constant) of the aircraft and the wind. The local wind components are obtained by calling the windfield, which is an input to the planner, at a given coordinate (latitude/longitude). The planner therefore accounts for wind by optimizing for time rather than distance.

B. Evaluation of Planner through Comparative Experiment

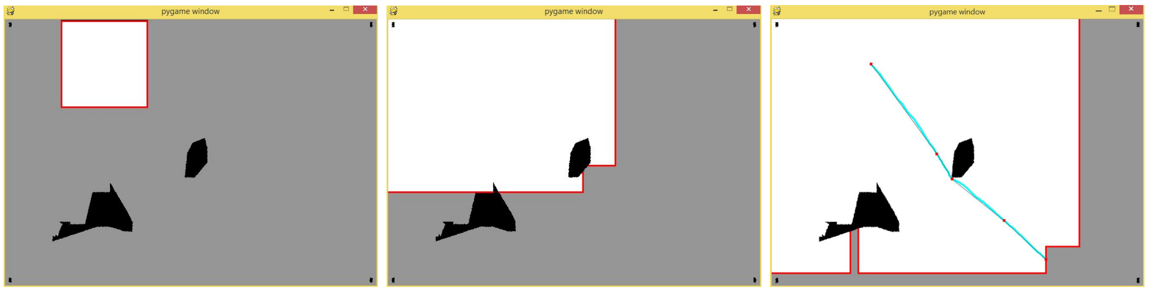
An input-output test bed platform is used to evaluate the branching planner. Therefore, the same input is given to both the branching planner and the benchmark planner, and their respective outputs are compared. In order to perform this comparative experiment, an implementation of the Girardet-based planner has been completed to serve as the benchmark planner and experiment variables have been defined. Brief descriptions of the implementation of the benchmark planner and the selection of the experiment variables are given here.

1. Benchmark Planner: Girardet's Method

An implementation of the planner based on Girardet's method has been completed by J. M. Hoekstra. Similar to the branching planner, this benchmark planner accepts as input an origin and destination (in latitude/longitude coordinates) and an obstacle field consisting of sets of obstacle vertices. The constant altitude and true airspeed of the aircraft are also defined as input. The windfield definition to be used can be changed within the algorithm code.

As described in Section I, the algorithm applied by Girardet relies on a discretization of the problem space. The dimensions defining this grid are therefore a necessary parameter to be defined for this planner. For the main experiment, grid dimensions will be held constant at an empirically defined value. The standard grid dimensions (701 x 501) have been chosen because the resulting rectangular mesh has a high enough resolution to produce solutions that consistently avoid obstacles. As grid dimensions are reduced from these standard granularities, the solution quality deteriorates (i.e. segments of the path may traverse an obstacle). The standard grid dimensions will therefore be maintained throughout the main experiment; however, additional grid resolution tests for the Girardet planner will be performed in order to contextualize the results of the experiment, especially with regard to computational cost.

In accordance with the discretization of the problem space, a moving front propagates outward from the origin, as shown in Figure 3a. The optimal route at each node is stored as the front traverses the node. Although the nodes are nominally spaced in regular intervals according to the selected discretization, an option for noise has been incorporated into the planner (and is activated for the experiment) to compensate for over-symmetry. Nodes contained within an obstacle have been identified so that the front propagates around obstacles, as shown in Figure 3b. When the front reaches the destination node, the front propagation is terminated and the optimal route will be displayed, as shown in Figure 3c.



(a) Moving front propagates outward from origin (b) Traverses nodes not contained within an obstacle (c) Optimal path recovered after front reaches destination

Figure 3: Animation of the benchmark planner's moving front

Since the optimal path returned by this benchmark planner is in a zigzag pattern due to the discretized space, an effort is made to smooth the path. The path definition is made less convoluted by minimizing the number of defining waypoints. It can be seen in Figure 3c that between the origin and destination there are three waypoints defined along the path. These are waypoints defining the linear approximations to almost linear segments of the original output of the planner. It is these waypoints that are logged as output to represent the solution obtained by the planner.

2. Independent Variables

The two parameters that will be varied are obstacle field complexity and windfield complexity. Three levels of the obstacle field complexity will be tested: a low, medium and high level. Similarly, three levels of the windfield complexity will be tested: a low, medium and high level.

Before attempting to quantify the levels of obstacle field complexity and windfield complexity, a discussion of the components that make an obstacle field or a windfield will be discussed. The creation of an obstacle field is meant to mimic reality. Therefore, the hazards or obstacles that an aircraft might need to avoid are considered for inclusion in the obstacle field variable for this study. Special Use Airspace (SUA) is selected as an obstacle to model, along with weather cells, such as thunderstorm areas, which are hereafter referred to as WX obstacles. Other aircraft were considered for inclusion in the obstacle field, but without the explicit modeling of time it is difficult to create a trajectory that avoids such dynamic obstacles. The SUA and WX obstacles are thus considered to be static obstacles and are modeled by polygons of sizes and geometries that represent reality, per available references.[18] SUA obstacles are characterized by more regular geometries than WX obstacles, whereas WX obstacles have many facets and are of a large size.

With regard to the windfield variable, direction and strength of wind is relevant to the aircraft. Therefore, for a certain area that represents the problem space of this study, winds will be defined by spatial interpolation between specifically defined points. The location of these points, and the number of these points plus their respective strengths and direction will influence the gradient of the wind. A wind definition in the "top left" of the problem space would have its greatest effect in the northeast region of the problem space; a strength between 90 and 150 knots is selected to represent what could reasonably be expected by an aircraft in cruise at about 37,000 feet. The wind direction represents the source of the wind, so a 90 degree wind is blowing from east to west. It was experimentally determined what combinations of wind specification location, direction and strength contributed to the complexity of the windfield. Even in the case that a low wind strength is used, for example, a highly complex windfield can be created through the placement and direction of the defined wind vectors.

A visualization of the independent variables and their levels is provided in Figure 4, which is followed by quantitative summaries in Tables 1 and 2.

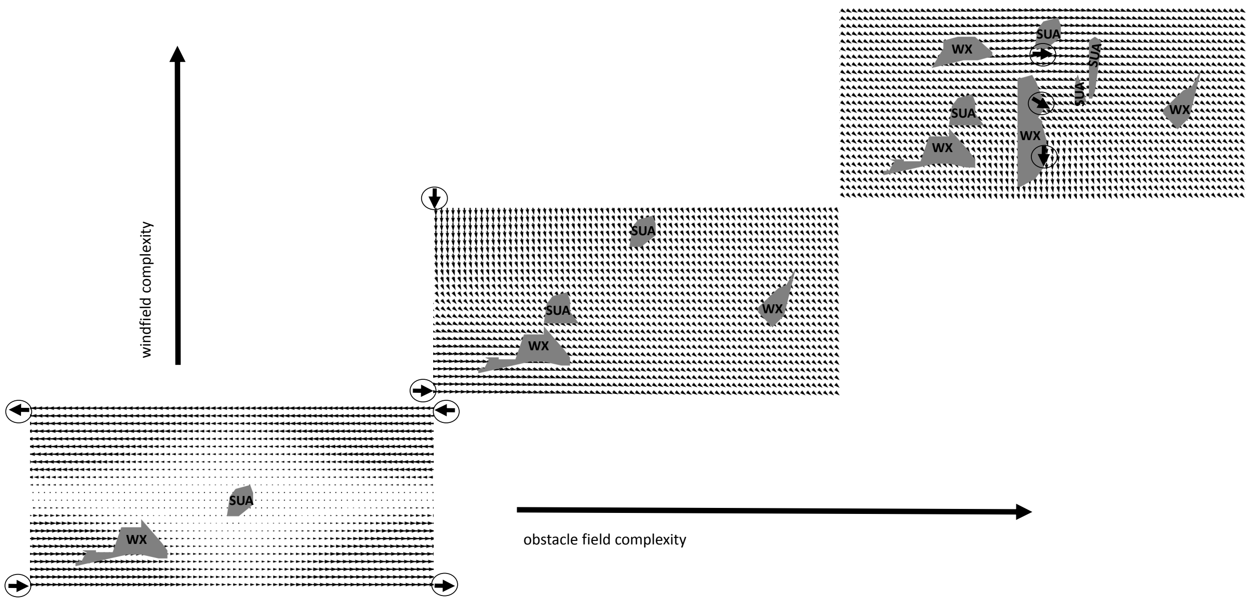


Figure 4: Levels of Independent Variables

The properties that define each independent variable level being used in this study are summarized in Tables 1 and 2.

Table 1: Levels of Obstacle Field Complexity

	Number of SUAs	Number of WX
High	4	4
Medium	2	2
Low	1	1

Table 2: Levels of Windfield Complexity

	Location	Direction (degrees)	Strength (kts)
High	Top Middle	270	90
	Middle Middle	300	90
	Bottom Middle	360	90
Medium	Top Left	360	150
	Bottom Left	270	150
Low	Top Left	90	150
	Top Right	90	150
	Bottom Left	270	150
	Bottom Right	270	150

As previously mentioned, a constant test area is used throughout all conditions. Figure 5 shows this test area, along with eight cities whose geographical coordinates are used as origins and destinations in this study. The city pairs are chosen such that eight directions are represented between origin and destination city; the order of the four city pairs therefore matters, and creates eight repetitions for the given condition being tested.

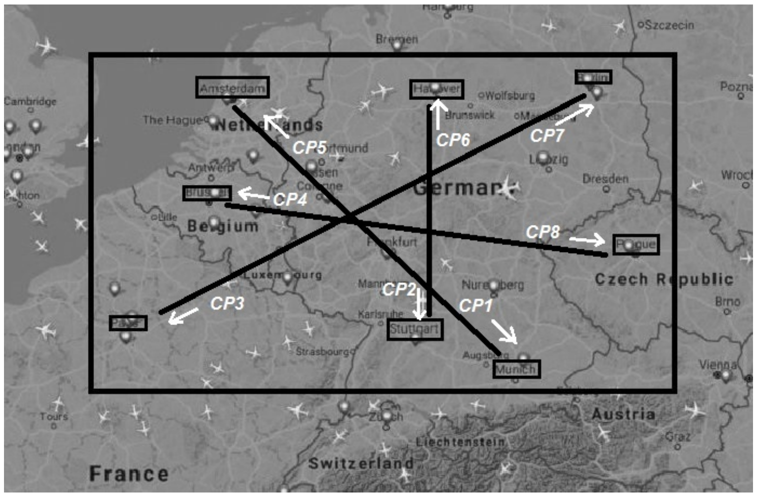


Figure 5: Constant area for all obstacle fields and windfields, eight repetitions of city-pair origins and destinations
(Background Image from FlightRadar24)

To further illustrate the independent variables of this study, an example test condition is shown in Figure 6. The obstacle field complexity is high, with 4 SUA and 4 WX obstacles. The windfield complexity is low, with a pair of winds being specified along the bottom of the test area traveling from west to east, and a pair of winds with an opposite direction along the top of the test area. It can be seen from the windfield quiver plot in Figure 6 that the wind streams become negated toward the center of the test area. The city pair in this example is “city pair 1”, hereafter shortened to CP1, representing travel from Amsterdam to Munich.

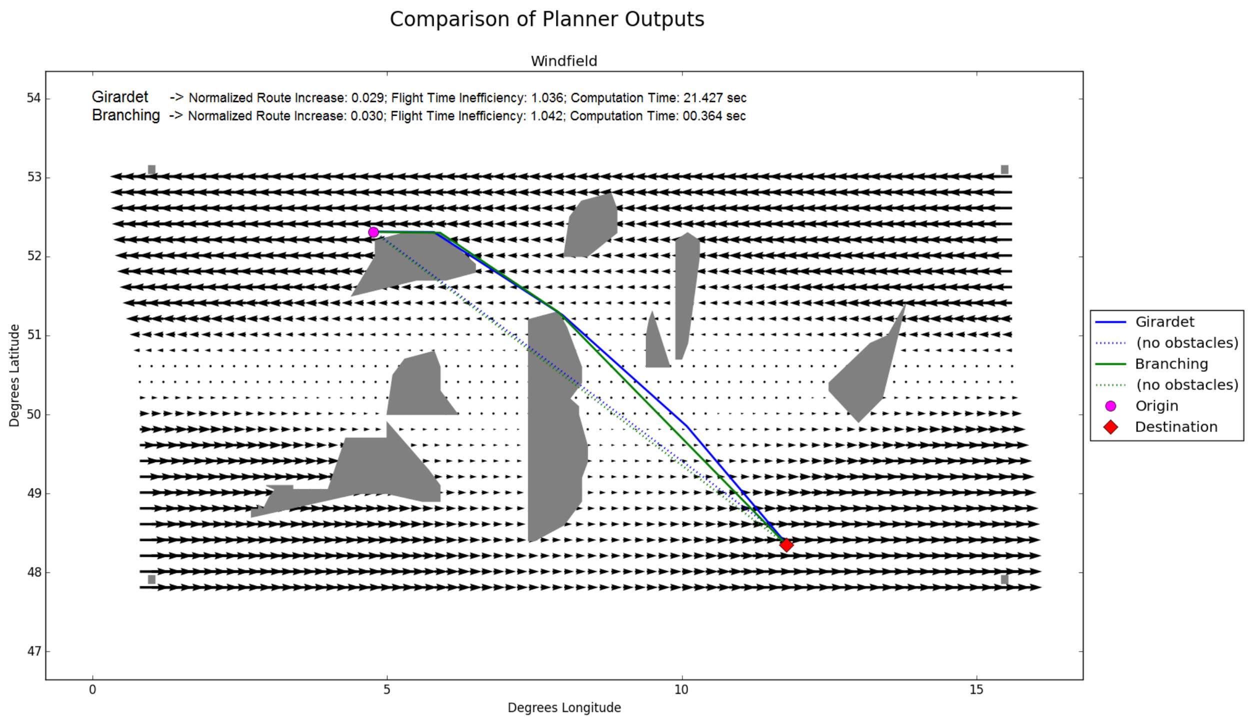


Figure 6: High Obstacle Field Complexity, Low Wind Field Complexity, Results for Repetition CP1

The resulting path outputs of the two planners are also plotted in Figure 6. Metrics for interpreting such results are explained in the following section.

3. Dependent Variables

In order to accept or reject the hypothesis stated in Section I, it is important to characterize the solution effectiveness and computation speed of the Branching planner and Girardet planner. Therefore, an effort has been made to develop metrics that define the effectiveness of a solution; the computation speed metric, on the other hand, is a simple runtime calculation.

The first indication of an effective solution is that the solution is feasible. A feasible solution in this study means that the path does not pass through any obstacles. A first metric will therefore be a 'yes' or 'no' answer to the following statement: a complete trajectory is generated and obstacles are avoided. If, for example, the planner encounters an error and cannot produce an output at all, or if the planner produces a path that intersects one or more obstacles, the result for this metric would be 'no'.

Further metrics are designed to answer the question of path effectiveness in terms of path deviation quantification. To approach this quantification logically, it is first considered that, in the absence of all obstacle and wind complexities, a preferred route by the flight management system (FMS) from point A to point B would follow the direct, shortest great-circle distance between those two points. It is thus assumed that an indicator for FMS acceptability would be the absence of extraneous waypoints that cause unnecessary changes to the navigation along the route. In an effort to model FMS acceptability, a metric that quantifies extra route length has been developed. In any of the test conditions, the extra route length can be calculated by comparing the planner's final path to the path that it would have created had there been no obstacles in the way. The difference in these path lengths can then be normalized by the 'no obstacle' path length. It is therefore necessary, in order to compute this metric, to run the test condition twice; once in the presence of the obstacle field and once without obstacles. The windfield and origin-destination pair remains the same across these two runs. To better understand what the results of these two runs for one test condition would look like, it is suggested to refer again to Figure 6. The solutions produced by the Branching and Girardet planners are plotted with solid lines, while the solutions that the planners would produce in the same situation but without obstacles are plotted with dotted lines. Such output needs post-processing to calculate the respective extra route lengths. To calculate the path length of either the actual solution or the ‘no obstacle’ solution for a given planner, the great-circle distances between each pair of waypoints along the path are calculated using the haversine formula (spherical earth assumption) and added together. The sum, or the total path length of the ‘no obstacle’ solution is then subtracted from the total path length of the actual solution; this value is normalized by the ’no obstacle’ length to generate the non-dimensional metric currently being discussed. This metric will hereafter be referred to as the normalized route increase metric.

A final metric to quantify the path deviations in terms of added flight time is developed. It is important to the authors to understand the efficiency, or rather inefficiency, of the solution in terms of flight time. A ratio between flight times of the planner’s output in the presence of obstacles and in the absence of obstacles is therefore included. Similar to the normalized route increase metric, this inefficiency metric will make use of the path definitions obtained by performing the same condition twice, once with obstacles and once without, to obtain the planner’s actual solution and the planner’s ‘no obstacle’ solution. The flight time for each of those paths is obtained in a similar manner as path length was obtained for the previous metric. Specifically, the flight times of each segment along the path are added together to determine the path’s total flight time. Flight time for a given segment is defined by the great-circle distance and calculated groundspeed. Groundspeed is obtained for a given path segment by incorporating the average wind experienced along that segment with the known true airspeed through vector decomposition. In this way, a total flight time is calculated for both the actual and the ‘no obstacle’ solutions output by each planner on a given condition. The planner’s inefficiency metric is thus the ratio of its actual solution to its ‘no obstacle’ solution.

With regard to the computation speed metric, runtime is calculated using the clock function of the Python time module. It should be noted that any visualizations provided by the planners are disabled for the experiment, and the entirety of the experiment is conducted on a HP ZBook 15 Mobile Workstation.

A summary of the dependent measures being used in this study is provided in Table 3 below.

Table 3: Summary of Dependent Measures

	Solution Effectiveness			Computation Speed
		(Path Deviations)		
Metric	complete trajectory generation with obstacles avoided	normalized route increase [-]	flight time inefficiency [-]	elapsed runtime [sec]
Equation	(YES or NO)	$\frac{distance_{actual} - distance_{no\ obstacle}}{distance_{no\ obstacle}}$	$\frac{flight\ time_{actual}}{flight\ time_{no\ obstacle}}$	(runtime)

A final comment on the dependent measures summarized in Table 3 is to highlight that the path deviation metrics are both relative to the planner’s own ‘best’ solution. It will also be possible to compare absolute flight time performance, for example, but the relative comparisons produced by these path deviation metrics are expected to offer further insight into the power and limitations of each planner.

III. Results

The results presented in this section are organized by dependent measure and are therefore divided into two categories: path comparisons and computational cost comparisons. Within each category, a similar structure is followed. First, the data is presented in a table showing the Branching planner and Girardet planner results for the given metric. Descriptive statistics are provided along with the data. Second, a further statistical analysis is described and supplemented with box plot figures.

In order to address all dependent measures of the experiment, it should be noted here before proceeding that both planners successfully generated a full trajectory on each run.

A. Path Comparisons

1. Normalized Route Increase Metric

The normalized route increase results are provided in Table 4 below. The results of each planner are organized in a 3x3 grid reflecting the levels of windfield complexity and obstacle field complexity. Within each cell, the non-dimensional route increase is reported for all repetitions. The eight repetitions within a cell are organized by value in descending order; each value is associated with a city pair number (i.e. CP1-CP8) to maintain traceability. The average normalized (i.e. dimensionless) route increase per cell is reported in bold.

Table 4: Normalized Route Increase Results

Branching

(windfield complexity)	Low			Medium			High		
	CP4	0.027		CP4	0.014		CP4	0.276	
	CP1	0.012		CP8	0.014		CP3	0.188	
	CP5	0.012		CP7	0.012		CP8	0.053	
High	CP8	0.001	**0.007**	CP3	0.009	**0.006**	CP1	0.030	**0.075**
	CP3	0.000		CP1	0.000		CP5	0.030	
	CP7	0.000		CP2	0.000		CP7	0.019	
	CP2	0.000		CP5	0.000		CP2	0.000	
	CP6	0.000		CP6	0.000		CP6	0.000	
	CP8	0.027		CP4	0.014		CP4	0.076	
	CP5	0.012		CP8	0.014		CP8	0.053	
	CP1	0.012		CP3	0.012		CP1	0.030	
Medium	CP4	0.001	**0.007**	CP7	0.009	**0.006**	CP5	0.030	**0.029**
	CP3	0.000		CP1	0.000		CP7	0.019	
	CP7	0.000		CP2	0.000		CP3	0.019	
	CP2	0.000		CP5	0.000		CP2	0.000	
	CP6	0.000		CP6	0.000		CP6	0.000	
	CP7	0.097		CP4	0.148		CP4	0.204	
	CP8	0.027		CP7	0.095		CP7	0.188	
	CP5	0.012		CP8	0.014		CP8	0.053	
Low	CP1	0.012	**0.019**	CP3	0.012	**0.034**	CP1	0.030	**0.066**
	CP4	0.001		CP1	0.000		CP5	0.030	
	CP3	0.000		CP2	0.000		CP3	0.019	
	CP2	0.000		CP5	0.000		CP2	0.000	
	CP6	0.000		CP6	0.000		CP6	0.000	

(obstacle field complexity)

Girardet

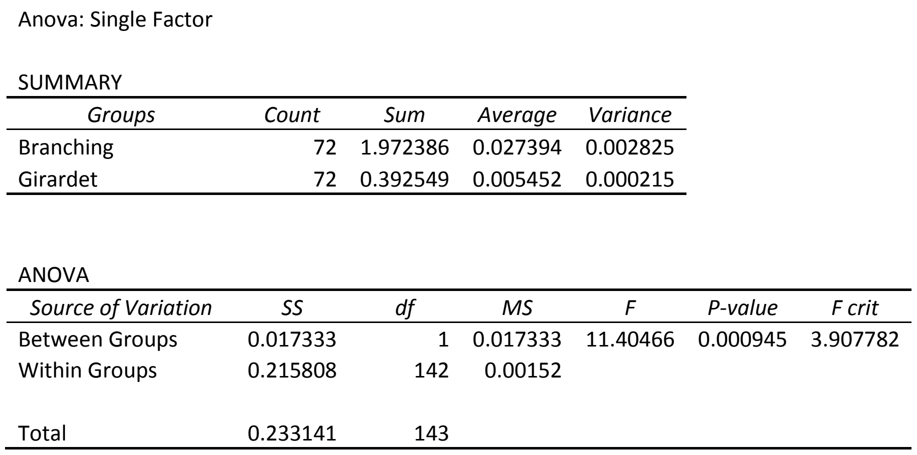
(windfield complexity)	Low			Medium			High		
	CP5	0.012		CP4	0.012		CP4	0.051	
	CP1	0.011		CP3	0.008		CP1	0.033	
	CP8	0.010		CP5	0.001		CP5	0.030	
High	CP4	0.003	**0.001**	CP1	-0.002	**-0.002**	CP3	0.019	**0.013**
	CP3	0.001		CP2	-0.003		CP6	0.001	
	CP6	-0.001		CP6	-0.003		CP2	-0.002	
	CP2	-0.006		CP8	-0.008		CP8	-0.010	
	CP7	-0.020		CP7	-0.018		CP7	-0.014	
	CP4	0.018		CP8	0.011		CP8	0.047	
	CP5	0.013		CP6	0.005		CP4	0.046	
	CP1	0.011		CP7	0.004		CP5	0.032	
Medium	CP8	0.000	**0.005**	CP4	0.003	**0.003**	CP1	0.030	**0.022**
	CP7	0.000		CP5	0.002		CP7	0.017	
	CP2	-0.001		CP3	0.002		CP3	0.012	
	CP3	-0.003		CP1	-0.001		CP2	-0.002	
	CP6	-0.003		CP2	-0.001		CP6	-0.004	
	CP1	0.013		CP7	0.007		CP1	0.029	
	CP2	0.010		CP5	0.001		CP5	0.027	
	CP5	0.008		CP6	0.000		CP7	0.015	
Low	CP7	0.000	**-0.001**	CP1	-0.001	**-0.003**	CP8	0.012	**0.010**
	CP6	-0.005		CP8	-0.006		CP2	0.007	
	CP8	-0.007		CP2	-0.007		CP4	0.003	
	CP4	-0.011		CP4	-0.010		CP6	0.000	
	CP3	-0.017		CP3	-0.010		CP3	-0.011	

(obstacle field complexity)

By comparing the normalized route increase averages, reported in bold in each cell of Table 4, it is apparent that for both planners (Branching and Girardet), this metric markedly increases between the low/medium and the high obstacle field complexities (i.e. first/second vs. third table column). Differences in windfield complexity (differences between table rows) have less of an obvious impact on the normalized route increase metric for either planner. In particular, for the Branching planner from medium to high windfield complexity there is no change in normalized route increase metric in two cases — the low and medium obstacle field complexities (i.e. first and second table columns).

The optimization priority for both planners is time rather than distance. However, this normalized route increase metric is useful in clarifying the behavior of each planner in a 'no obstacle' situation. The metric is computed using the path length of the planner's solution to the same problem but without obstacles. The baseline for the Branching planner is a direct route while the baseline for the Girardet planner is a wind-optimal route. It therefore makes sense that the Girardet planner is capable of producing negative values for the normalized route increase metric.

An analysis of variance (ANOVA) is preformed in order to state whether there is a significant difference between the two distributions formed by all normalized route increase metrics for the Branching planner and for the Girardet planner. The ANOVA presented in Figure 7 shows a p-value less than 0.05 and therefore suggests there is a significant difference between the Branching planner and the Girardet planner for this metric. Nevertheless, a perhaps more informative figure, Figure 8, shows that several outliers could be accountable for the result of the ANOVA.



Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
Branching	72	1.972386	0.027394	0.002825
Girardet	72	0.392549	0.005452	0.000215

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.017333	1	0.017333	11.40466	0.000945	3.907782
Within Groups	0.215808	142	0.00152			
Total	0.233141	143				

Figure 7: ANOVA for Normalized Route Increase Results

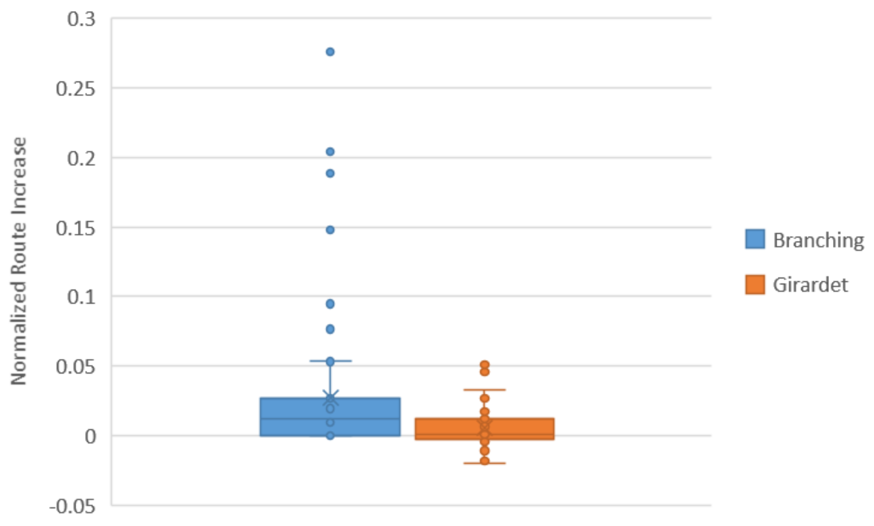


Figure 8: Normalized Route Increase Box Plot

2. Flight Time Inefficiency Metric

The flight time inefficiency data is provided in Table 5. Note that values provided in each cell of the table are dimensionless, as they represent a ratio of flight times.

Table 5: Flight Time Inefficiency Results

Branching

(windfield complexity)									
	CP4	1.020		CP8	1.020		CP4	1.270	
	CP1	1.016		CP4	1.016		CP3	1.146	
	CP7	1.008		CP7	1.008		CP8	1.047	
High	CP8	1.003	**1.006**	CP3	1.006	**1.006**	CP5	1.039	**1.066**
	CP5	1.002		CP1	1.000		CP2	1.028	
	CP2	1.000		CP2	1.000		CP1	1.015	
	CP6	1.000		CP5	1.000		CP7	1.001	
	CP3	0.999		CP6	1.000		CP6	0.982	
	CP7	1.039		CP7	1.068		CP7	1.124	
	CP5	1.009		CP4	1.023		CP4	1.092	
	CP8	1.009		CP8	1.010		CP8	1.051	
Medium	CP4	1.005	**1.005**	CP1	1.000	**1.010**	CP1	1.029	**1.039**
	CP2	1.000		CP2	1.000		CP5	1.028	
	CP6	1.000		CP5	1.000		CP6	1.003	
	CP1	0.992		CP6	1.000		CP2	1.000	
	CP3	0.983		CP3	0.982		CP3	0.988	
	CP7	1.128		CP4	1.151		CP7	1.239	
	CP8	1.021		CP7	1.131		CP4	1.215	
	CP5	1.010		CP8	1.012		CP8	1.052	
Low	CP1	1.008	**1.019**	CP3	1.005	**1.037**	CP1	1.042	**1.074**
	CP4	1.001		CP1	1.000		CP3	1.021	
	CP2	1.000		CP2	1.000		CP5	1.017	
	CP6	1.000		CP5	1.000		CP2	1.004	
	CP3	0.989		CP6	1.000		CP6	1.004	
	Low			Medium			High		

(obstacle field complexity)

Girardet

(windfield complexity)									
	CP1	1.023		CP4	1.010		CP4	1.064	
	CP8	1.013		CP5	1.004		CP5	1.053	
	CP6	1.011		CP6	0.993		CP3	1.022	
High	CP5	1.009	**0.997**	CP2	0.991	**0.989**	CP1	1.019	**1.006**
	CP4	1.008		CP1	0.989		CP6	1.009	
	CP3	0.992		CP3	0.986		CP2	0.998	
	CP2	0.968		CP8	0.981		CP8	0.944	
	CP7	0.949		CP7	0.961		CP7	0.936	
	CP4	1.024		CP8	1.005		CP4	1.048	
	CP5	1.019		CP6	1.005		CP7	1.038	
	CP2	1.001		CP5	1.003		CP8	1.038	
Medium	CP8	0.999	**1.003**	CP4	1.000	**1.000**	CP1	1.033	**1.024**
	CP7	0.998		CP2	1.000		CP5	1.030	
	CP3	0.995		CP3	0.999		CP3	1.006	
	CP6	0.995		CP1	0.996		CP6	0.998	
	CP1	0.993		CP7	0.992		CP2	0.997	
	CP5	1.014		CP7	1.024		CP1	1.036	
	CP2	1.011		CP3	1.003		CP5	1.021	
	CP1	1.010		CP6	1.001		CP3	1.018	
Low	CP7	1.007	**1.002**	CP1	0.999	**1.000**	CP7	1.016	**1.014**
	CP3	0.996		CP5	0.996		CP8	1.012	
	CP6	0.995		CP8	0.993		CP2	1.005	
	CP8	0.994		CP2	0.992		CP6	1.003	
	CP4	0.987		CP4	0.991		CP4	0.999	
	Low			Medium			High		

(obstacle field complexity)

The averages presented in bold in Table 5 suggest a similar performance between planners with respect to the flight time inefficiency metric. Although an ANOVA across all runs suggests that there is a significant difference, the boxplot again summarizes the statistical findings with more clarity.

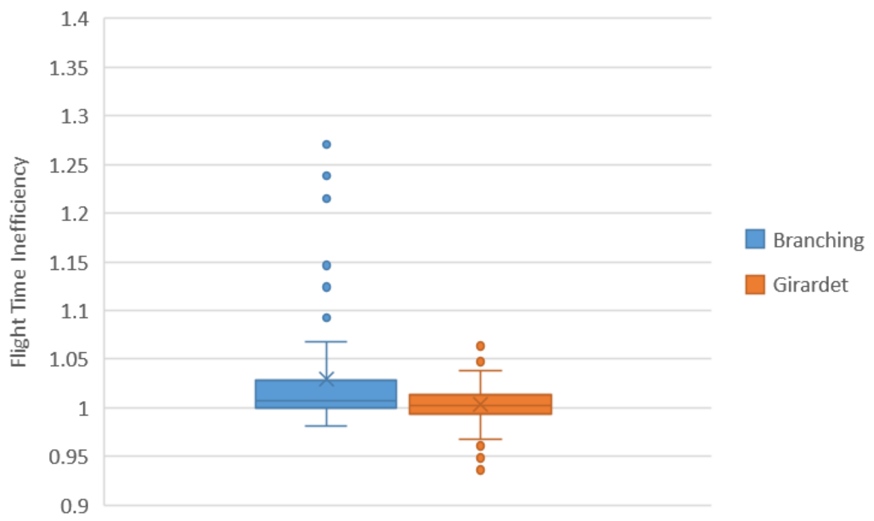


Figure 9: Flight Time Inefficiency Box Plot

Due to the importance of this metric and its bearing on the study, further statistical analysis has been conducted on each cell within Table 9. A single factor ANOVA carried out between planners for each cell yields a p-value greater than 0.05 for all but one cell. The only condition showing a significant difference between planners for the flight time inefficiency metric is the 'high windfield complexity, medium obstacle field complexity' condition with a p-value of 0.013.

Further analysis related to this metric has been conducted in anticipation of further discussion in Section IV. The absolute flight times of the solutions obtained by the Branching and Girardet planners have been extracted and are summarized in Figure 10.

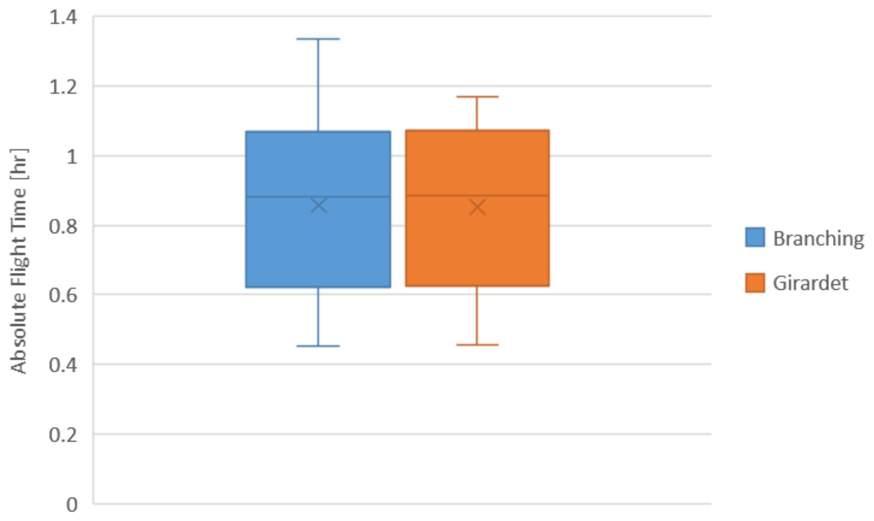
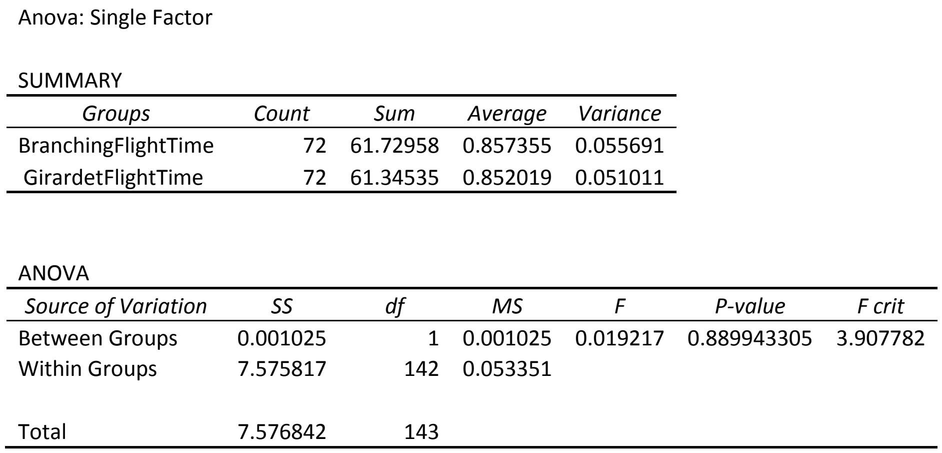


Figure 10: Absolute Flight Time Box Plot

The box plot shown in Figure 10 shows that across all 72 runs, the solutions of the Branching and Girardet planners yield comparable flight times. A further statistical inference can be made with an ANOVA, as shown in Figure 11.



Anova: Single Factor

SUMMARY

Groups	Count	Sum	Average	Variance
BranchingFlightTime	72	61.72958	0.857355	0.055691
GirardetFlightTime	72	61.34535	0.852019	0.051011

ANOVA

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	0.001025	1	0.001025	0.019217	0.889943305	3.907782
Within Groups	7.575817	142	0.053351			
Total	7.576842	143				

Figure 11: ANOVA for Absolute Flight Time Results

Given that the p-value reported in Figure 11 is far greater than 0.05, it may be concluded that there is no significant difference between absolute flight time distributions created by the solutions of the two planners.

B. Computational Cost Comparisons

The computation time, in seconds, for each planner is provided in Table 6. One row of the Girardet planner data in Table 6 is highlighted to identify the conditions that have been re-tested with varying grid resolutions. As previously mentioned, the additional grid resolution tests for the Girardet planner are performed in order to contextualize the computational cost metric. These additional results are thus presented after the main-experiment results shown in Table 6 and Figure 12.

Table 6: Computation Time [sec] Results

Branching

(windfield complexity)	Low			Medium			High		
	CP1	0.392		CP1	0.432		CP3	0.482	
	CP5	0.392		CP3	0.432		CP4	0.461	
	CP7	0.374		CP6	0.394		CP8	0.435	
High	CP3	0.347	**0.353**	CP7	0.367	**0.378**	CP7	0.431	**0.413**
	CP8	0.334		CP2	0.361		CP5	0.414	
	CP4	0.331		CP8	0.361		CP1	0.373	
	CP2	0.327		CP4	0.340		CP2	0.369	
	CP6	0.325		CP5	0.335		CP6	0.336	
	CP1	0.841		CP1	0.705		CP4	0.811	
	CP3	0.561		CP2	0.665		CP3	0.726	
	CP8	0.510		CP4	0.467		CP8	0.486	
Medium	CP7	0.425	**0.475**	CP8	0.445	**0.477**	CP1	0.460	**0.527**
	CP4	0.392		CP5	0.414		CP5	0.456	
	CP6	0.373		CP6	0.378		CP7	0.455	
	CP2	0.356		CP7	0.374		CP6	0.426	
	CP5	0.338		CP3	0.364		CP2	0.397	
	CP6	0.616		CP8	0.686		CP8	0.436	
	CP2	0.370		CP4	0.398		CP7	0.429	
	CP1	0.369		CP7	0.358		CP3	0.423	
Low	CP7	0.361	**0.384**	CP6	0.344	**0.390**	CP5	0.415	**0.399**
	CP3	0.358		CP5	0.343		CP2	0.396	
	CP4	0.335		CP3	0.336		CP4	0.375	
	CP8	0.333		CP1	0.329		CP1	0.364	
	CP5	0.332		CP2	0.327		CP6	0.351	

(obstacle field complexity)

Girardet

(windfield complexity)	Low			Medium			High		
	CP3	23.766		CP3	23.578		CP3	22.266	
	CP8	23.551		CP8	23.310		CP8	22.207	
	CP7	23.071		CP7	22.403		CP7	21.502	
High	CP6	22.283	**22.095**	CP6	21.699	**21.436**	CP1	21.103	**20.759**
	CP1	21.713		CP4	20.624		CP5	19.958	
	CP2	21.038		CP2	20.617		CP6	19.834	
	CP4	20.740		CP1	20.139		CP4	19.635	
	CP5	20.596		CP5	19.121		CP2	19.564	
	CP3	23.737		CP3	23.255		CP3	22.066	
	CP8	23.688		CP8	22.786		CP8	21.791	
	CP7	22.428		CP7	22.040		CP1	21.190	
Medium	CP6	22.355	**22.111**	CP6	21.311	**21.155**	CP7	21.156	**20.527**
	CP1	21.907		CP4	20.479		CP6	19.600	
	CP4	21.087		CP2	20.170		CP5	19.573	
	CP2	21.026		CP1	19.921		CP4	19.458	
	CP5	20.660		CP5	19.274		CP2	19.381	
	CP8	23.816		CP8	23.366		CP3	22.233	
	CP3	23.792		CP3	23.156		CP8	22.178	
	CP7	22.686		CP7	22.456		CP1	21.427	
Low	CP6	21.985	**22.102**	CP6	21.981	**21.450**	CP7	21.413	**20.848**
	CP1	21.602		CP4	20.764		CP2	20.040	
	CP4	21.189		CP2	20.627		CP4	19.867	
	CP2	21.133		CP1	19.914		CP5	19.863	
	CP5	20.610		CP5	19.335		CP6	19.766	

(obstacle field complexity)

From the average runtimes reported in Table 6, it is apparent that the Branching planner is two orders of magnitude faster than the Girardet planner with standard grid resolution. A single factor ANOVA gives a p-value less than 0.05, suggesting a significant difference between the results of the two planners for this metric. The box plot in Figure 12 provides further evidence of this difference.

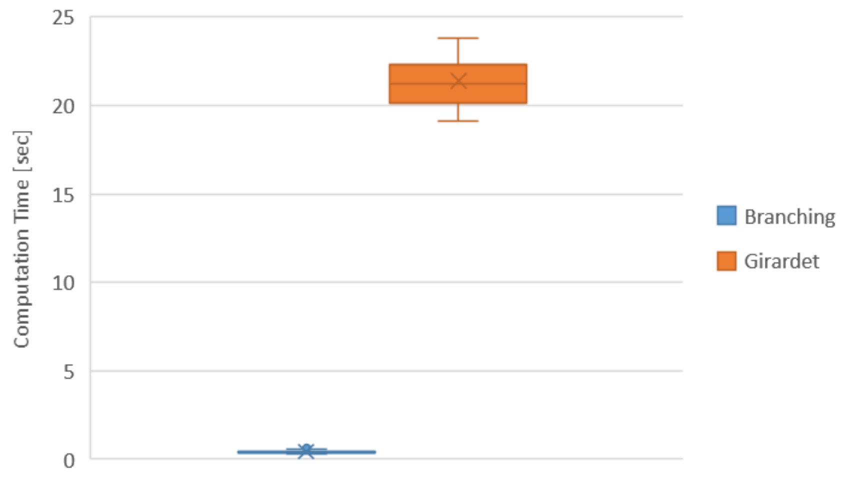
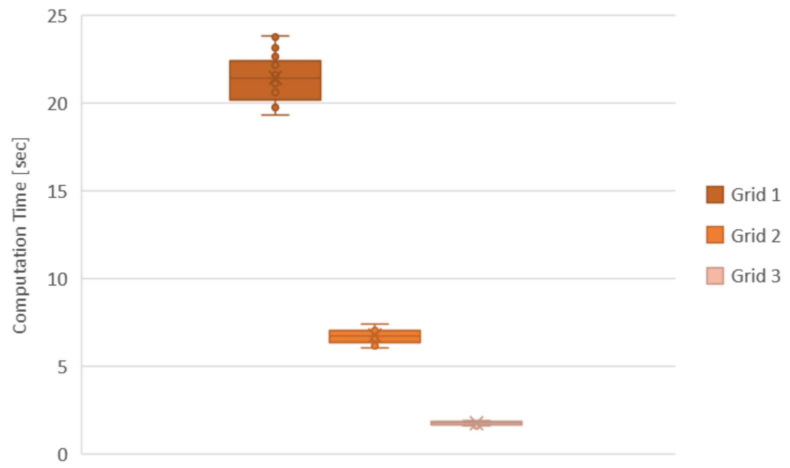
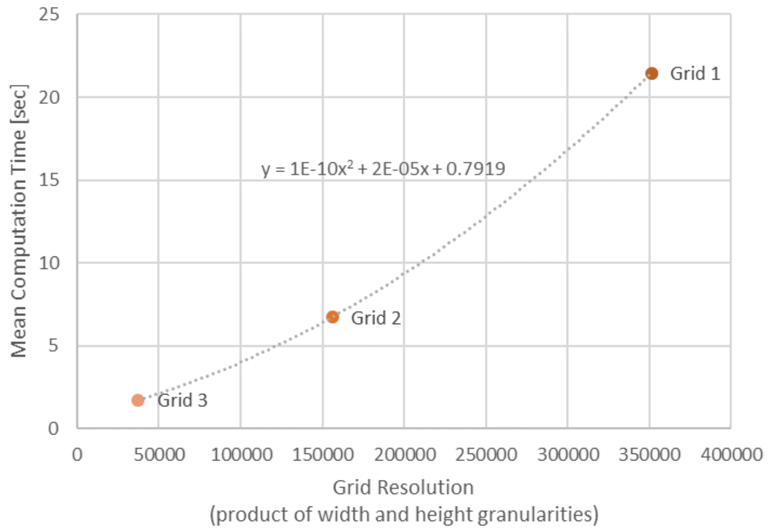


Figure 12: Computation Time Box Plot

In order to better understand the influence of grid resolution on computation time, the highlighted subset of Girardet data in Table 6 has been re-tested with different grid resolutions. In Figure 13, the resolution represented by Grid 1 is the same as the one used throughout the main experiment, i.e. the resolution resulting from grid dimensions 701 x 501. Grids 2 and 3, on the other hand, have "lower" resolutions because they cover the same area as Grid 1 but have grid dimensions 467 x 334 and 224 x 167, respectively. To clarify, the term "grid dimensions" refers to the grid's width granularity and height granularity, and the product of these granularities is used as a quantification of grid resolution.



(a) Grid Resolution Box Plot



(b) Computation Time as a Function of Grid Resolution

Figure 13: Girardet Planner: Grid Resolution Effect on Computation Time

Figure 13a clearly shows a relationship between Girardet planner computation time and grid resolution. The computation time decreases as grid resolution is reduced. To further investigate this relationship, Figure 13b shows a scatter plot of mean computation time per tested grid resolution. A polynomial regression offered the best fit for the data; this curve and its equation are included in Figure 13b. The leading term of the regression equation indicates that computation time depends on the square of the grid resolution, which is comparable to the time complexity of Dijkstra's algorithm.

IV. Discussion

This section includes a discussion of the results presented in Section III and the implications of these results. The results of the path comparison measures are discussed first, followed by the results of the computational cost metric. The section concludes with more general considerations of relevance to the study.

The first of two path comparison measures is the normalized route increase metric. The data given in Table 4 raises several points, which will be discussed here. First, there are values of 0.000 representing route increase for the Branching planner. This occurs when a direct route exists between the city pair and there are no obstacles to interfere. Although the optimization priority is for time rather than distance, the Branching planner is initialized with the direct route and will only create new trial solutions if the direct route is obstructed by an obstacle. Therefore, in the cases where the results for this metric have a value of 0.000, the direct route is unobstructed by obstacles and becomes the final solution, regardless of wind-optimality. Second, there are several values of normalized route increase that occur more than once in the Branching portion of Table 4. It is not surprising that the path produced by the Branching planner for a certain iteration is reproduced in either **a)** an iteration of the same condition where the city pair origin and destination are switched or **b)** an iteration of a different condition but with the same city pair and the absence of any wind/obstacle which causes a path adjustment (due, for example, to the preference of a different rotation about an obstacle). A third observation regarding the normalized route increase metric is the ability for the Girardet planner to generate negative values. Since the 'no obstacle' path generated by the Girardet planner (which is used to calculate this metric) is a wind-optimal route, rather than a distance-optimal route, it might have a longer path length than the path length of the solution to the 'with obstacles' problem.

The normalized route increase metric was designed to model FMS acceptability; a longer route length is assumed to be less favorable for an FMS since it is likely to involve more changes in navigational direction. However, it should be noted that the Girardet output being analyzed is actually a simplification of the more convoluted route produced by the optimal control algorithm. Additional route lengths inherent to the Girardet planner's original output have been reduced through smoothing and are therefore not reflected by the outcome of this metric.

The second of two path comparison measures is the flight time inefficiency metric. The authors find this metric to be more significant for the study than the previous metric, especially since the optimization priority is for time rather than distance. In considering the data presented in Table 5, it should be noted that values indicating an inefficiency less than 1 seem unreasonable. However, the small discrepancies can be explained by approximations used to obtain this dependent measure. The flight time for any segment along a route has been approximated by a combination of true airspeed and average wind for that segment. Therefore, this metric does suffer from the potential to accumulate errors due to approximation across route segments. The same method for obtaining the inefficiency metric, however, is used for the output of both planners, so a bias in the results is not expected. It is notable that the averages presented in Table 5 are characterized by very little variance. The box plot shown in Figure 9 reveals similar median values for the Branching and Girardet data sets, but it is apparent that outliers will affect an ANOVA. A comparison of absolute flight times rather than inefficiency metric helps to situate the data. It becomes clear through Figure 10 that the absolute flight times have similar averages but that the Branching planner data produces an upper whisker characterized by greater flight time. This result is to be expected, as the Girardet planner essentially exhausts all possible routes with the moving front, while the Branching planner will only create a new route option upon encountering an obstacle.

The computational cost metric seemingly offers the most decisive result in terms of comparing the Branching and Girardet planners. However, several points must be considered to properly interpret the computation time results. A single constant grid size, for example, was used for the Girardet planner throughout the experiment that produced the results reported in Tables 4 through 6. As previously mentioned, this grid size was empirically found such that the quality of the Girardet planner solutions was acceptable. Therefore, a separate investigation has been conducted to understand the effect of the Girardet planner's grid resolution on runtime. The results of the investigation offer context for the computation time results provided in Table 6. This context shows that the Girardet planner computation time varies with the square of the grid resolution, meaning that this implementation of the planner has a similar time complexity to that of Dijkstra's algorithm, or $\mathcal{O}(N^2)$. Although making the Girardet grid more coarse significantly reduces runtime, it also reduces the solution quality such that the smoothed route intersects obstacles. It is therefore suggested that, in the continuation of this research, discretizations other than the regular grid used in this experiment be considered. For example, a grid that is created strategically in response to the positions of the obstacles may provide the benefit of fewer nodes, thereby reducing runtime, while maintaining solution quality.

In general, the difference between methods tested in this study has been highlighted by the results. An illustration to give a further example of these results is given in Figure 14. In this condition, with low obstacle field complexity, a fundamental difference between planners is highlighted. The Branching planner solution will by definition remain close to obstacles since the driver for trajectory generation is branching on encountered obstacles. The Girardet planner, on the other hand, traverses the entire solution space and is therefore capable of producing a solution that strays far from any obstacle. This benefit becomes most influential in the high windfield complexity cases, where large gains can be made from a particularly well-chosen route. In a situation similar to the one depicted in Figure 14, the advantage of the Girardet planner is clearly illustrated. It is in the high obstacle field complexity cases, however, that the Girardet planner may suffer from terminating its search immediately upon the front arriving at the destination: potential routes may not have been discovered if the front did not fully encircle all obstacles before termination. Perhaps the introduction of more noise into the grid to indirectly affect the movement from one node to another would decrease the chances of this missed opportunity from happening. However, the introduction of more noise into the grid used by the Girardet planner would also diminish the already limited transparency of the planner's behavior. To summarize, the Girardet planner shows its greatest advantage in the high windfield complexity case and perhaps a main limitation with regard to obstacle fields in general.

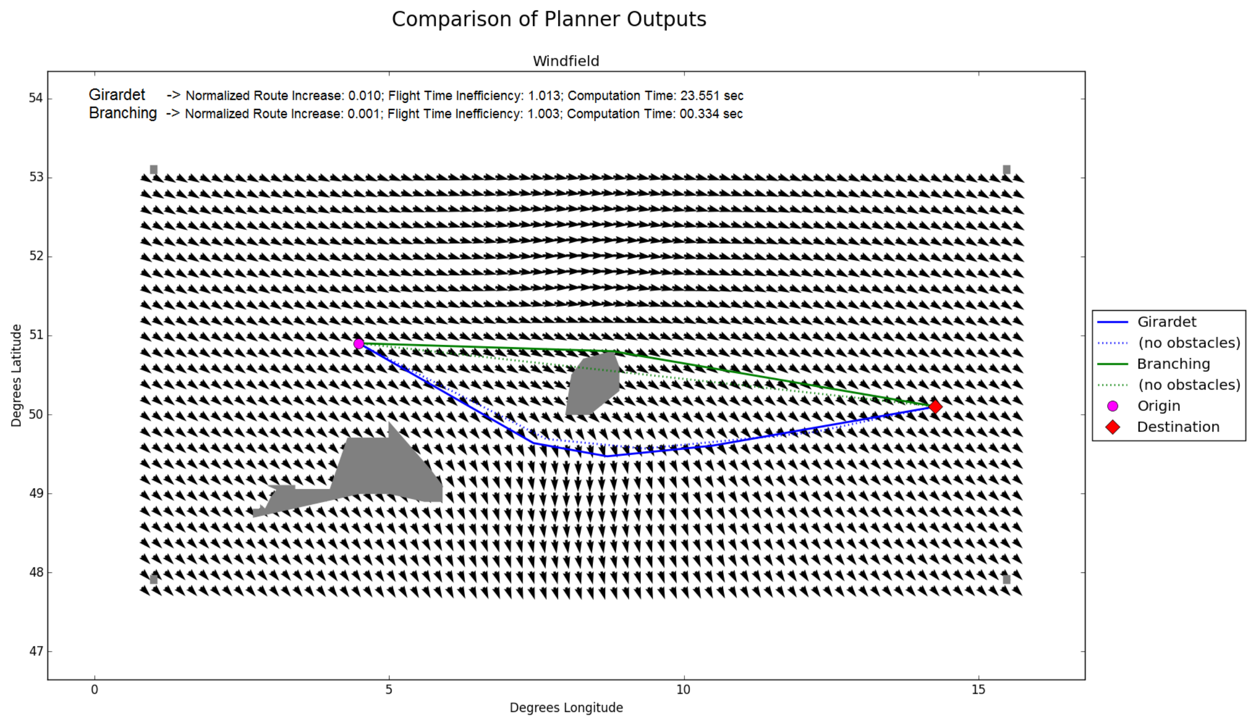


Figure 14: Low Obstacle Field Complexity, High Wind Complexity Results for Repetition CP8

It is therefore suggested that in the continuation of this research, a separation of tasks between the Girardet planner and the Branching planner be investigated. In other words, the advantages of each planner can be capitalized upon by using the Girardet planner for finding the wind-optimal route and subsequently using the Branching planner to avoid obstacles along this initial route option. The overall Girardet method includes the final de-congestion step using simulated annealing, which requires a lengthy computation time; perhaps the areas identified by Girardet's congestion metric could instead be modeled as obstacles in and of themselves for the Branching planner logic to handle. This sequencing might align particularly well with the operational demands on the planners. For example, it is envisioned that the use of the Branching planner can be generalized to include traffic as an obstacle. In a TBO environment, it is feasible that the location of traffic at given times in the future would be known. These predicted obstacle locations, however, may only be known with a shorter planning horizon than the wind forecast, for example, since aircraft trajectory requests would likely depend on weather conditions. With the wind forecast being known further in advance, there is adequate time for the Girardet planner to generate a wind-optimal route that can then be passed to the Branching planner.

V. Conclusion

The feasibility of a heuristic branching planner for use in four-dimensional trajectory (4DT) planning has been validated by this study. The heuristic branching method-based planner, otherwise referred to as the Branching planner, that was developed for this study yielded results that were comparable to that of the benchmark Girardet planner but at a significantly lower computational cost. The normalized route increase metric generated several outliers for both planners, which made it difficult to test for statistical significance between distributions. However, the flight time inefficiency metric showed that the data distributions produced by the Branching planner and Girardet planner were statistically insignificant from each other for all but one condition. A statistical test of all data for absolute flight time yielded no significant difference. The runtime metric shows a significant reduction in computation time for the Branching planner, compared to the Girardet planner. Thus, the results of this study suggest no reason to reject the hypothesis. It is concluded that further research into the application of a heuristic branching planner to 4DT operations in the air traffic management realm is warranted.

As stated in the discussion of the previous section, the researchers see a potential for combining the strengths of the two planners by providing the 'no obstacle' wind-optimal route output from the Girardet planner as input to the Branching planner for the necessary obstacle avoidance adjustments. Both planners are currently limited to a constant altitude, and traffic has yet to be included in the obstacle field that would be considered by the Branching planner. It is expected that the logic of the Branching planner will generalize nicely for the addition of altitude, specifically by considering options for maneuvering above and below obstacles, just as clockwise and counterclockwise options have been considered. For the Girardet planner, however, the addition of altitude will mean extending the algorithm to another dimension, which promises to be a more difficult task. In terms of including traffic as an obstacle, it is expected that the Branching planner will handle traffic as it handled the SUA and WX obstacles of the current study. It may also be worth considering the inclusion of constraints on dynamics, such as those limitations known to the flight management system, in the Branching planner algorithm such that output will be dynamically feasible, especially when small obstacles such as traffic are considered. By addressing these limitations, the potential contributions and detriments of the heuristic branching method, as applied to 4DT planning, will be more clearly understood.

Acknowledgments

The authors would like to thank the members of the CNS/ATM group for their valuable insights and suggestions which contributed to this work.

References

[1]U.S.-EU MOC Annex 1 - Coordination Committee, "NextGen/SESAR State of Harmonisation Document," 2014.

[2]Dalmau, R. and Prats, X., "How much fuel can be saved in a perfect flight trajectory? Continuous cruise climbs vs. conventional operations," *6th International Congress on Research in Air Transportation (ICRAT)*, Istanbul, 2014.

[3]Joint Planning and Development Office, "Concept of Operations for the Next Generation Air Transportation System Version 2.0," June 2007.

[4]Da Silva, S., "Trajectory-Based Operations(TBO)," *ICAO SIP 2012 - ASBU workshops*, Bangkok, 2012.

[5]LaValle, S. M., *Planning Algorithms*, Cambridge University Press, 2006.

[6]Girardet, B., "Trafic Aérien: Détermination optimale et globale des trajectoires d'avions en présence de vent," 2014, Ph.D. Dissertation.

[7]Baras, J. and Theodorakopoulos, G., *Path Problems in Networks*, Synthesis Lectures on Communication Networks #3, Morgan & Claypool Publishers, Breinigsville, Pennsylvania, 2010.

[8]Hwang, Y. K. and Ahuja, N., "A Potential Field Approach to Path Planning," *IEEE Transactions on Robotics And Automation*, Vol. 8, No. 1, 1992, pp. 23–32.

[9]Girardet, B., Lapasset, L., Delahaye, D., Rabut, C., and Brenier, Y., "Generating Optimal Aircraft Trajectories with respect to Weather Conditions," *2nd International Conference on Interdisciplinary Science for Innovative Air Traffic Management*, Toulouse, 2013.

[10]Sethian, J. A. and Vladimirsky, A., "Ordered Upwind Methods for Static Hamilton-Jacobi Equations: Theory and Algorithms," *Society for Industrial and Applied Mathematics*, Vol. 41, No. 1, 2003, pp. 325–363.

[11]Sethian, J. A., "Ordered Upwind Methods," 2010, Accessed 1 July 2016.

[12]Rios, J. and Lohn, J., "A Comparison of Optimization Approaches for Nationwide Traffic Flow Management," *AIAA Guidance, Navigation, and Control Conference*, Chicago, August 2009.

[13]International Civil Aviation Organization, "Global Air Navigation Plan for CNS / ATM Systems," 2002, Doc 9750.

[14]Hillier, F. S. and Lieberman, G. L., *Introduction to Operations Research*, McGraw-Hill International Edition, 10th ed., 2011.

[15]Karlof, J. K., editor, *Integer Programming: Theory and Practice*, Taylor & Francis Group, Boca Raton, 2006.

[16]Eele, A. and Richards, A., "Path-Planning with Avoidance Using Nonlinear Branch-and-Bound Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 2, 2009.

[17]Upadhyay, S. and Ratnoo, A., "Smooth Trajectory Planning for MAVs with Airspace Restrictions," *AIAA Guidance, Navigation, and Control Conference*, AIAA SciTech, San Diego, 2016.

[18]Eurocontrol and Federal Aviation Administration (FAA), "2015 Comparison of Air Traffic Management-Related Operational Performance: U.S./Europe," 2016.

Part II

Book of Appendices

Appendix A

Literature Review

A-1 Research Context

As stated in Part I, the air transportation systems of the United States and Europe are facing significant challenges under increased demand. Next Generation Air Transportation System (NextGen) and Single European Sky ATM Research (SESAR), in an effort to accommodate this demand, seek to achieve similar goals. In specific, the 2014 NextGen-SESAR State of Harmonisation Document states that "The goals on each side of the Atlantic are to improve overall aviation system performance, particularly in the areas of flight efficiency and environmental impact, while also meeting expected demands for increased capacity and continuing to maintain the highest levels of safety" (U.S.-EU MOC Annex 1 - Coordination Committee, 2014, pp. 8). A major Air Traffic Management (ATM) concept in both NextGen and SESAR is Trajectory-Based Operations (TBO), which meets the aforementioned goals. According to the International Civil Aviation Organization (ICAO), TBO is slated for approval in 2028 (Da Silva, 2012).

Further details about TBO, including associated benefits and challenges, are discussed in the following section. Also included in this chapter is a section on trajectory planning considerations; the material of this chapter provides the background information that is necessary in order to interpret the literature survey of Section A-2 and to motivate the proposed research.

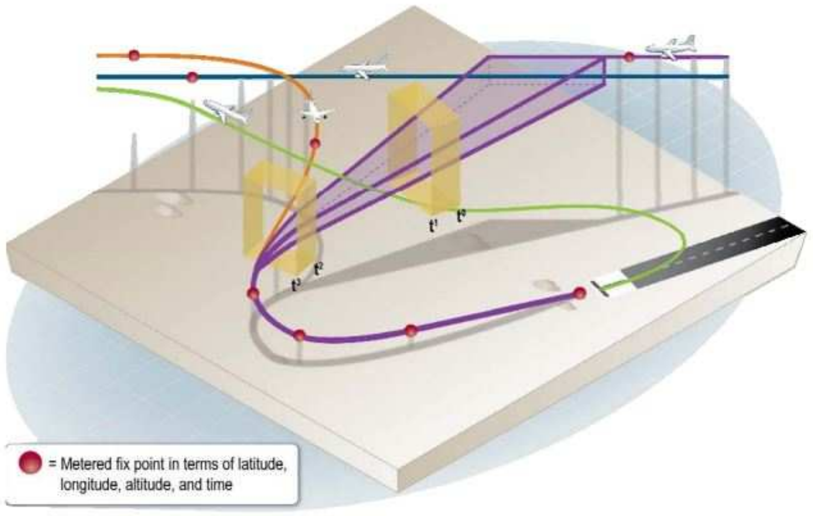
A-1-1 Trajectory Based Operations

In an effort to provide a working definition of TBO, an early definition of Trajectory Operations from the Federal Aviation Administration (FAA) is referenced: "the concept of an air traffic management system in which every aircraft that is operating in or managed by the system is represented by a Four-Dimensional Trajectory (4DT)" (Ashford, 2010, pp. 5). A 4DT is a trajectory defined in 3D-space (latitude, longitude, altitude) as well as time. However, there are nuances between the definitions of 4DT that are envisioned by NextGen and by SESAR (Enea & Porretta, 2012). To illustrate, Figure A-1 shows 4DT schematics from NextGen and SESAR, respectively. The differences between the two representations in Figure A-1 do not necessarily indicate incongruities between the two visions; instead, they shed light on emphasized elements of each vision. For example, the NextGen representation (Figure A-1a) includes uncertainties in position and time. As highlighted in Enea and Porretta's publication, the "flight-operating environment" is the entity that drives the "required level of specificity," in terms of buffers and tolerances along the 4DT (Enea & Porretta, 2012, pp. 2). There may be a range of times, for example, over which an aircraft can cross particular waypoints and still make a downstream crossing restriction. The Controlled Time of Arrivals (CTAs) paired with certain waypoints in a 4DT specify these allowable 'windows' in time (Enea & Porretta, 2012). An analysis of SESAR's representation (Figure A-1b), on the other hand, reveals the emphasis of a gate-to-gate 4DT that has been negotiated by "all the ATM stakeholders"; this agreed-upon 4DT is called the Reference Business Trajectory (RBT) (Enea & Porretta, 2012, pp. 3)

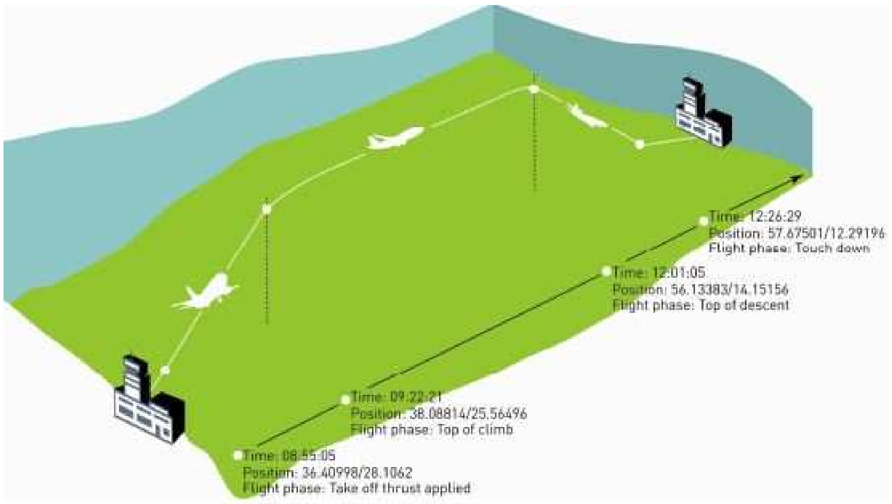
Regardless of the subtle distinctions between the NextGen and SESAR definitions of 4DT, both visions include a system-wide sharing of the trajectory information (Enea & Porretta, 2012).

The shift from present-day operations to TBO, i.e. from clearance-based to performance-based operations, will yield many benefits. For example, an expected improvement of TBO over the current system is an increase in capacity due to a reduction in trajectory uncertainty (Joint Planning and Development Office, 2007). ATM under TBO will no longer suffer from inefficiencies of generic predefined routes, but will cater toward capabilities of individual aircraft. Not only is efficiency addressed in this way, but also environmental impact. For example, continuous climbs and descents (enabled by 4DT operations) offer significant reductions in fuel and emissions (Dalmau & Prats, 2014). In a study that compares continuous cruise climbs to conventional operations, Dalmau and Prats evaluate trajectories that have been optimized between just after takeoff and just before landing to trajectories that are constrained to fly at a constant cruise altitude (Dalmau & Prats, 2014). The goal of the study is to quantify the savings of flying optimized profiles, since a reduction in fuel consumption, and thus emissions, has been identified as "one of the main concerns of the different aviation stakeholders" (Dalmau & Prats, 2014). The authors observe that, along with fuel consumption, flight time is reduced by continuous operations; this result is significant, since fuel consumption and time are usually inversely related (Dalmau & Prats, 2014).

The 4DT provides the basis for separation management decisions in TBO (Joint Planning and Development Office, 2007). Since intent is clearly stipulated in the trajectory definition and the trajectory definition is shared with all ATM stakeholders, the imminent shift toward TBO will result in a more proactive ATM system. For example, instead of reacting to traffic or other obstacles that arise as an aircraft is flying its flight plan, a trajectory under TBO takes into account anticipated conflicts and therefore shifts the operations into a more strategic rather than tactical realm. A major aspect of this future ATM concept relies on the ability to plan aircraft trajectories such that a) the efficiency of the individual flight is maximized while b) safety is ensured by avoiding interactions both between trajectories and also between trajectories and other potential hazards. A trajectory planner, therefore, is an enabling technology that is crucial to the viability of TBO in real life operations. Such a trajectory planner would be useful to airlines and could be used in an Airline Operations Center (AOC). The planner would also serve those in Air Traffic Flow Management (ATFM). Multiple users benefit from the 4D trajectory planner in TBO; as such, user preferences vary and can be accounted for in the planning by the user's input of a cost-index. The automated planner can therefore optimize a trajectory based on user input, which satisfies the previously mentioned a), while avoiding all foreseen obstacles to satisfy b). A survey of planning methods is provided in the next chapter; first, however, a section on planning considerations is included here.


t¹
t²
t¹
t²
= Metered fix point in terms of latitude, longitude, altitude, and time

(a) NextGen publication (Joint Planning and Development Office, 2011)


Time: 08:55:05
Position: 36.40998/28.1062
Flight phase: Take off thrust applied
Time: 09:22:21
Position: 38.08814/25.56496
Flight phase: Top of climb
Time: 12:01:05
Position: 56.13383/14.15156
Flight phase: Top of descent
Time: 12:26:29
Position: 57.67501/12.29196
Flight phase: Touch down

(b) SESAR publication (SESAR Joint Undertaking, 2010)

Figure A-1: NextGen and SESAR 4DT Schematics

A-1-2 Trajectory Planning Considerations

Enabling Technologies

Various technologies are required to make TBO feasible. A brief list of these enabling technologies is provided here (Enea & Porretta, 2012):

a) Advanced Flight Management System (FMS) Capabilities

b) Data Communication

c) Automatic Dependent Surveillance-Broadcast (ADS-B)

d) Air Traffic Control Decision Support Tools

The listed technologies are in various stages of maturity. A major challenge in making them fully operational, and thus in enabling TBO, is the issue of interoperability. As an example, trajectory data must be shared across various decision support tools, and these tools may be designed for systems that are incompatible with each other (Cate, 2013). In order to discover such issues, testing of new technologies is paramount and depending on the technology and its purpose, human-in-the-loop simulations or flight testing are two options for evaluating the technology.

Flight tests related to TBO have occurred recently in both the United States and in Europe. For a month in 2011, for example, Alaska Airlines participated in flight trials of the Required Time of Arrival (RTA) concept; 71% of the 595 participating aircraft fully executed an RTA into the Seattle terminal area (Wynnyk, Balakrishna, Macwilliams, & Becher, 2013). The FMS onboard the aircraft is central to the success of meeting an RTA and the ground automation is central to the success of creating the 4DT that drives that RTA (Wynnyk et al., 2013). Flight tests in Europe, under the Initial 4D Project (I4D) project, have tested the benefits of an initial TBO scheme in a mixed equipage environment (Wynnyk et al., 2013). These I4D flight tests, along with the TBO flight tests in the United States, highlight the importance of considering operational implications when designing a new technology.

Operational Implications

This subsection treats several representative studies in order to establish a general framework for considering the operational implications of potential algorithms. To gain a better understanding of what may be expected by users of a 4DT planner, studies involving first ground-based users and then airborne users are reviewed (Erzberger, 2006; Ballin, Sharma, Vivona, Johnson, & Ramiscal, 2002). Two examples of airborne systems that have reached high levels of maturity are reviewed for an insight into certification processes (Koczo & Wing, 2013; Bussink et al., 2016).

A conflict resolution tool for air traffic controllers has been developed by Erzberger at NASA Ames Research Center, and his work is discussed in a 2006 paper for the 25th International Congress of the Aeronautical Sciences (Erzberger, 2006). Erzberger's paper is reviewed here, both for the general structure of the conflict resolution tool and for its user-centric features.

The tool can be described as consisting of several modules: the Resolution Aircraft and Maneuver Selector (RAMS), the Resolution Maneuver Generator (RMG), the 4D Trajectory Synthesizer (TS) and the Conflict Detector (CD) (Erzberger, 2006, pp. 2). Information is passed between modules, either in linear progression or in iteration loops, to get from the input (a pair of aircraft in conflict) to the output (a conflict-free resolution trajectory) (Erzberger, 2006, pp. 2). The tool takes into account the typical preferences of air traffic controllers in solving particular types of conflicts; a lookup table of conflict type and preferred maneuver sequence is used (Erzberger, 2006, pp. 7). Preference is also given to those maneuvers that deviate the least from the original trajectory in order to accommodate users from the air side as well (Erzberger, 2006).

With regard to airborne air transportation system users, Ballin et al. offer insight into crew decision-making aids through the development of the Autonomous Operations Planner (AOP) (Ballin et al., 2002). The AOP is a decision support tool that takes into account “traffic flow management and airspace constraints, schedule requirements, weather hazards, aircraft operational limits, and crew or airline flight-planning goals” (Ballin et al., 2002, pp. 1). In an air traffic system with airborne separation standards (rather than ground-based separation), the AOP would be especially useful. Regardless of whether separation assurance is performed in the air or on the ground, however, the functionality of the AOP highlights the importance of generating trajectories that “are consistent with the FMS’s aircraft performance model, navigation database, crew preferences, cost function data, and constraint data” (Ballin et al., 2002, pp. 7). The FMS would therefore be a key technology (as discussed in Section A-1-2) in terms of cooperation with the 4DT planner to guide the decision of whether or not a particular trajectory option is flyable and thus viable from a flight deck perspective.

A description of the safety analysis and certification process involved with an Electronic Flight Bag (EFB) addition, namely the Traffic Aware Strategic Aircrew Requests (TASAR), is given by Koczo and Wing (Koczo & Wing, 2013). TASAR is meant to be a decision support tool for pilots to “identify and recommend” changes in flight trajectory that would be operationally beneficial and would likely be accepted by air traffic controllers (Koczo & Wing, 2013, pp. 1). An important characteristic of TASAR is that it is “advisory-only” and therefore has little to no effect in the event of failure, which is a consideration of the required safety analysis (Koczo & Wing, 2013, pp. 2). Koczo and Wing address the concern of overwhelming the air traffic controllers with requests by suggesting that the number of TASAR users will increase only gradually (Koczo & Wing, 2013, pp. 6). Relevant regulatory documents published by the Federal Aviation Administration of the United States are listed in Koczo and Wing’s work, along with the five certification phases; this information could serve as a beneficial reference for future 4DT planner research.

Another example of a flight deck technology in a mature stage of development is Time and Energy Managed Operations (TEMO); the results of a human-in-the-loop study testing the technology were reported by Bussink et al. this year (Bussink et al., 2016). TEMO is an optimization tool for the vertical flight trajectory and is consistent with Continuous Descent Operations but additionally uses “time-metering at two control points to facilitate flow management and arrival spacing” (Bussink et al., 2016, pp. 1). The technology was tested by pilots in a six-degree-of-freedom motion simulator to showcase a Technology Readiness Level 5 status (Bussink et al., 2016, pp. 1).

Trajectory Interactions

To further investigate the operational framework for the 4DT planner, a few practical considerations in terms of trajectory interactions will be detailed in this section.

The separation minimums illustrated in Figure A-2 are considered standard separation for en-route operations. Therefore, it will be a requirement of the 4DT planner to avoid traffic with the Closest Point of Approach (CPA) being at least 5 nautical miles (NM) in the lateral direction or at least 1000 feet in the vertical direction.

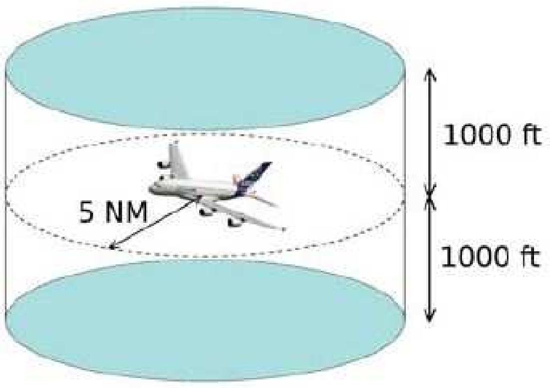


Figure A-2: Standard separation in en-route airspace (Girardet, 2014)

Besides traffic, the other obstacles an aircraft must avoid can be categorized as either meteorological phenomena or airspace/terrain restrictions[1]. Section A-2-1 includes the important statement that all obstacles can be defined as polygons. Therefore, it is considered to be the case that all phenomena an aircraft must avoid can be modeled as a polygon with enough buffer added to the definition of the obstacle such that nearest miss requirements for any obstacle are met.

A-2 Survey of Planning Methods

This chapter contains a literature review with the aim of answering the following question: which algorithms are relevant to aircraft trajectory planning, and what are their associated advantages and disadvantages in terms of computational and operational viability in the ATM domain? Literature has been divided into several main topic groups; these topics will be treated in separate sections of the chapter. The sections are: trajectory planning with Unmanned Aerial Vehicle (UAV) applications (Section A-2-1), shortest path algorithms (Section A-2-2), and stochastic search methods (Section A-2-3), followed by a dedicated section for the key reference of this research project (Section A-2-4).

[1]For further details, the first two appendices of Girardet's Ph.D. dissertation provide useful references for airspace and aviation meteorology details (Girardet, 2014)

A-2-1 Trajectory Planning: UAV Applications

For inspiration regarding algorithms that could be applied to the aircraft trajectory planning problem, literature within the UAV domain has been closely reviewed. Since UAV navigation is based on autonomous operations, the underlying methods of UAV navigation could provide invaluable insight into how the 4DT trajectory planner might work.

This year, researchers from Zhejiang University in China reported on their work to create a conflict-free 4D trajectory for multiple UAVs (Yang, Fang, & Li, 2016). Yang and co-authors recognize two categories of 4D trajectory generation methods: the "mathematical expression to describe the position-time relation of a trajectory" and "graph search techniques" (Yang et al., 2016, pp. 84). These "conventional" methods are deemed "inappropriate" for UAVs for two reasons (Yang et al., 2016, pp. 84). First, the methods usually involve 3D trajectories with added time restrictions, rather than 4D trajectories in and of themselves. Second, the produced trajectory definitions are often "complex". Due to the discontinuity created by using 3D trajectories and the excessive computational power required, these conventional methods seem to Yang and co-authors to be a non-viable option. Instead, the researchers use a bio-inspired method: the tau theory, originally authored by Lee in *Perception* in 1976. Yang's work uses a generalized definition of the variable tau, which is a "ratio of the distance and the relative movement velocity between the animal and its target" (Yang et al., 2016, pp. 85). Since this particular research application requires multiple UAVs to maneuver, the tau theory is considered together with harmonic motion to generate a guidance strategy. Trajectories are generated from this strategy and then optimized using a modified Particle Swarm Optimization (PSO) technique. While harmonic motion and PSO may not be relevant for the planning of a single aircraft's trajectory, the work by Yang and colleagues does clarify the distinct groups of "conventional" methods, as well as the availability of an alternative bio-inspired method.

An example of the first of two conventional methods identified by Yang et al. is the mathematically defined trajectory used by Upadhyay and Ratnoo from the Indian Institute of Science. In a report in January of this year, Upadhyay and Ratnoo described the "Four Parameter Logistic Curve (4PL) curve" as a potential solution to the problem of "generating smooth trajectories in a bounded airspace having arbitrar[ily] shaped no-fly zones and obstacles" (Upadhyay & Ratnoo, 2016, pp. 1). The authors compared the 4PL method to "Bézier, B-spline and clothoid curves" and found the generated trajectory to be smooth and less computationally complex (Upadhyay & Ratnoo, 2016, pp. 2). For the trajectory to be smooth is a major consideration, since UAVs cannot follow non-continuous trajectories. The 4PL is a logistic curve defined by four parameters: the initial and final ordinates of the curve and two curve-shaping parameters. Trajectories based on the 4PL curve can be "S" shaped or "half-S" shaped, and are defined by closed-form equations that can be proven to avoid recognized obstacles. Aside from the convenient table of comparisons for these mathematical trajectory definition methods, the work of Upadhyay and Ratnoo also reminds the reader that "any arbitrar[ily] shaped obstacle can be modeled as [a] convex polygon" (Upadhyay & Ratnoo, 2016, pp. 7). This property enables a researcher to consider all obstacles in a similar manner.

To consider the second of the two conventional methods for 4D trajectory generation, a comparison of graph-search methods will be reviewed. For this review, the work done by Sathyaraj et al. to compare "Dijkstra's algorithm, Bellman Ford's algorithm, Floyd-Warshall's algorithm and the AStar algorithm" is invaluable (Sathyaraj, Jain, Finn, & Drake, 2008, pp. 257). Sathyaraj et al. select the previously mentioned algorithms on the basis of being applicable to path planning for multiple UAVs. These minimum path algorithms are "heuristic search strategies [that] originated in the artificial intelligence (AI) field" (Sathyaraj et al., 2008, pp. 259). With respect to the "number of searches required and the time taken for computation," the AStar algorithm is deemed the best performer out of the four algorithms considered in this work (Sathyaraj et al., 2008, pp. 259). All algorithms make use of a weighted graph, but the AStar algorithm, according to the authors, combines the strengths of both a heuristic approach and a formal approach. While the AStar guarantees the shortest path, the heuristic upon which the algorithm is based does not in and of itself guarantee the shortest path. A property that is common among all algorithms in the informative comparison by Sathyaraj and co-authors is that computational time increases with the number of nodes in the graph.

The following section of this review provides a more detailed examination of the recently mentioned shortest path algorithms.

A-2-2 Shortest Path Algorithms

The shortest path problem involves a graph. A graph is defined as ordered pairs of vertices and edges, while a path is "an ordered sequence of vertices, such that an edge exists between two successive vertices" (Baras & Theodorakopoulos, 2010, pp. 1). The shortest path from one particular vertex to another is the path between those vertices with the "smallest path weight" (Baras & Theodorakopoulos, 2010, pp. 2). Weights are given to edges according to an appropriate "real-valued edge weight function" (Baras & Theodorakopoulos, 2010, pp. 2). The weight w of a path p (with vertices v_0 through v_k) is given in Equation A-1 below (Baras & Theodorakopoulos, 2010, pp. 2).

$$w(p) = \sum_{i=0}^{k-1} w(v_i, v_{i+1}) \tag{A-1}$$

The three algorithms (other than AStar) studied by Sathyaraj et al. and discussed in the previous section of this review are designed to compute the minimum path weight; the algorithms can be augmented, however, to compute the associated path as well (Baras & Theodorakopoulos, 2010, pp. 3). The algorithms differ in the paths that are iterated upon to arrive at the shortest path weights (Baras & Theodorakopoulos, 2010, pp. 3). Dijkstra's algorithm solves the shortest path problem when all edges have non-negative weights with a time complexity of $O(n^2)$ (Baras & Theodorakopoulos, 2010, pp. 3-4). Bellman-Ford can solve the problem with negatively weighted edges, but has a time complexity of $O(n^3)$ (Baras & Theodorakopoulos, 2010, pp. 4). Floyd-Warshall solves the shortest path problem from every vertex (not just from one source) with a time complexity of $\Theta(n^3)$ (Baras & Theodorakopoulos, 2010, pp. 5).

As the number of vertices (or nodes) in the graph increases, any of the three algorithms becomes computationally expensive at a drastic rate and thus less operationally viable. However, an alternative solution is to "consider using some kind of enumeration procedure" that is "cleverly structured so that only a tiny fraction of the feasible solutions actually need be examined," such as the branch-and-bound method (Hillier & Lieberman, 2011, pp. 501). The branching portion of the branch-and-bound method is to subdivide the "feasible solution" set and the bounding portion is to define for each subset "how good its best feasible solution can be" (Hillier & Lieberman, 2011, pp. 503). The branch-and-bound method enables a "shortest path" to therefore be found without iterating through all combinations of nodes. An example of this method being applied to the trajectory planning of a vehicle, with obstacle avoidance, is the work of Eele and Richards; this work is especially informative with empirical data for nine branching strategies (Eele & Richards, 2009). The order in which to process sub-problems created by branching is defined by the chosen search strategy; two examples of search strategies are the best-first search and the depth-first search (Karlof, 2006). In the case of the best-first search, the number of sub-problems to consider is minimized, thus the method is relatively quick in improving the lower bound (Karlof, 2006, pp. 269). This method has great potential for the 4DT planner application.

A-2-3 Stochastic Search Methods

For nonlinear optimization problems, stochastic search methods are often useful. Two such methods, genetic algorithms and simulated annealing, are considered to be "metaheuristics" that incorporate both global searches and local optimization (Hillier & Lieberman, 2011). To better understand these examples of metaheuristics and their applications to aircraft trajectory planning, Uçan's work with genetic algorithms and Chaimatanan's work with simulated annealing are considered (Uçan & Altilar, 2012; Chaimatanan, Delahaye, & Mongeau, 2012).

The research of Uçan and Altilar from 2012 is focused on the application of genetic algorithms to the problem of "navigation planning" in changing environments (Uçan & Altilar, 2012, pp. 1). The authors recognize the conventional route planning algorithms studied by Sathyaraj et al. that were introduced in subsection A-2-1 of this review. However, they argue that dynamic environmental constraints invalidate these methods (Uçan & Altilar, 2012, pp. 2). Uçan and Altilar assert that even dynamic path planning algorithms, such as "Frigioni, Franciosa, and Ramalingam Reps" will not be adequate when multiple aspects of the environment change at once (Uçan & Altilar, 2012, pp. 14). These algorithms are nonetheless compared to the proposed genetic algorithm: the analytical algorithms are inferior to the genetic algorithm, which "finds better results in a shorter computation time" (Uçan & Altilar, 2012, pp. 14). Such results are largely dependent on the evolutionary algorithm parameters, such as "selection method" and "mutation rate," which have been manipulated in an experiment (Uçan & Altilar, 2012, pp. 8). In a similar way, the simulated annealing algorithm which will be discussed next depends on several researcher-defined parameters.

The research of Chaimatanan, Delahaye, and Mongeau (also from 2012) highlights the potential use of Simulated Annealing (SA) in an ATM setting (Chaimatanan et al., 2012). The simulated annealing method uses an analogy with "annealing in solids" to provide "a framework for optimization of the properties of very large and complex systems"; this method was originally described by Kirkpatrick in 1983 (Kirkpatrick, Gelatt, & Vecchi, 1983, pp. 671). The search strategy of simulated annealing is influenced by the cooling schedule, which in the case of Chaimatanan's work should be a slow and smooth reduction in temperature to avoid convergence to a local minimum (Chaimatanan et al., 2012, pp. 7). A local search may be added, however, to the global-search-oriented SA settings in order to reduce computation time. Chaimatanan and co-authors conclude that the addition of a greedy search method to their SA procedures "reduces significantly the computing time" (Chaimatanan et al., 2012, pp. 9).

To bolster the conclusions of the two works just presented, a comparative study by Rios and Lohn concludes that between integer programming, simulated annealing, and genetic algorithms, it is the latter two stochastic search methods that are "indifferent in terms of runtime to the difficulty of any given scenario" with respect to the traffic flow management problem (Rios & Lohn, 2009, pp. 10). However, Rios and Lohn also state that integer programming would likely be "the tool of choice" due to its superior "solution quality and runtime" (Rios & Lohn, 2009, pp. 9). Even though the integer programming method is "limited to linear objective functions," the work done by Rios and Lohn in comparing this method to simulated annealing and genetic algorithms suggests that these last two metaheuristic methods can be outperformed (Rios & Lohn, 2009, pp. 10).

A-2-4 Method Published by Girardet

The work of Girardet, particularly the doctoral dissertation of 2014, is an essential component of this review and provides a key reference for the proposed research (Girardet, 2014). To introduce the algorithm and results that are central to Girardet's dissertation, an earlier publication by the same author is reviewed in the next subsection before delving into greater detail in Section A-2-4.

Introduction to the Ordered Upwind Algorithm

The relevance of the Ordered Upwind algorithm to this review is made clear by Girardet's work from 2013, titled "Generating optimal aircraft trajectories with respect to weather conditions" (Girardet et al., 2013). Girardet presents the Ordered Upwind algorithm, originally developed by Sethian and Vladimirsky, as "avoid[ing] iterations through a careful use of the information about the characteristic directions of the PDE" (Girardet et al., 2013, pp. 5). At the basis of this method is a Hamilton-Jacobi equation that can be formulated as a "front propagation problem" (Sethian & Vladimirsky, 2003, pp. 336). Although the Ordered Upwind method involves steps that are similar to Dijkstra's algorithm, it "converges towards the exact solution of the Hamilton-Jacobi equation" (Girardet et al., 2013, pp. 6). The complexity of the Ordered Upwind method, which depends on the number of mesh points N and is $O(N \log N)$, is similar to that of the graph-search methods (Girardet et al., 2013).

Girardet includes results obtained using an Eulerian discretization to compute trial values required by the Ordered Upwind method (Girardet et al., 2013). The Semi-Lagrangian discretization could be applied as an alternative, but then iterative algorithms would become necessary to solve for the roots of a non-linear equation (Girardet et al., 2013, pp. 9). Figure A-3 reproduces the visual results included in Girardet's work, where optimal trajectories that take into account obstacles have been computed using the Eulerian discretization and the Ordered Upwind method (Girardet et al., 2013).

As Girardet has shown, wind has a significant influence over the optimal (time-saving) trajectory. An underlying assumption, therefore, would be that wind information would be available for use by a 4D trajectory planner. Work done by Reynolds and co-authors offers an insight

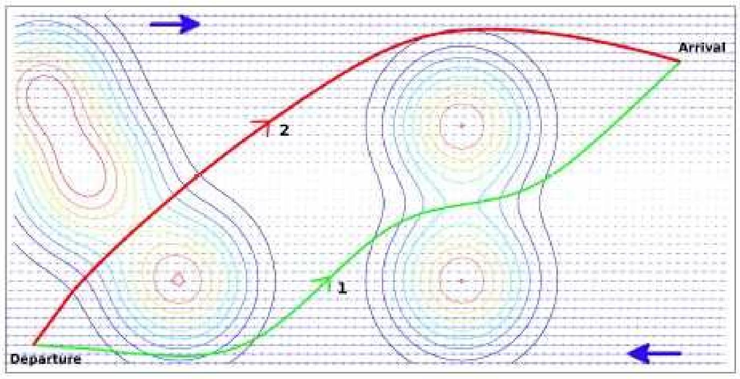


Figure A-3: Reproduction of Girardet's optimal trajectory computation results without wind (green) and with wind (red) (Girardet et al., 2013, pp. 10)

into "the relationship of wind information accuracy to 4D-TBO performance for a selection of operationally-relevant scenarios" (Reynolds, McPartland, Teller, & Troxel, 2015, pp. 10). Though the details of this work are considered outside the scope of this review's research aim, it is relevant to consider the operational implications of the potential algorithms. Such operational considerations, such as the certification prospects of selected algorithms, have been reviewed in subsection A-1-2.

Ordered Upwind Algorithm Definition

As mentioned in Section A-2-4, the optimal path problem considered by Girardet fits the form of a Hamilton-Jacobi equation. More specifically, the problem can be modeled by a static Hamilton-Jacobi equation by fixing the time of arrival, thereby removing the dependence on time (Girardet, 2014). A review of the key equations used to arrive at the Hamilton-Jacobi formulation will be given here before continuing with further details of the Ordered Upwind method.

The Hamilton-Jacobi equation is derived from a combination of the Bolza formulation of the original optimal control problem and the principle of optimality from dynamic programming, which is a discrete time method that makes use of Bellman's notion that an optimal solution contains the optimal solutions of all sub-problems (Girardet, 2014). The Bolza formulation is as follows (Girardet, 2014, Eqn. 2.8):

$$J(x(.), a(.), t_f) = \int_{t_0}^{t_f} g(t, x(t), a(t))dt + l(t_f, x(t_f)) \tag{A-2}$$

In Eqn. A-2 above, the initial time (t_0) is fixed and the final time (t_f) is unknown. The term being integrated is an instantaneous cost while the final term of Eqn. A-2 is the cost associated with the end point. It should be clarified that Girardet has defined the parameters of the problem at hand in the following way (Girardet, 2014, pp. 58):

$$\begin{cases} J(x(t), a(t)), & \text{the criterion to be optimized: total time of the trajectory} \\ x(t), & \text{state: position of the aircraft} \\ a(t), & \text{control: direction of displacement} \\ & \text{dynamics: the equation for aircraft velocity (dependent on control)} \\ & \text{initial conditions: begin at starting point} \\ & \text{final conditions: finish at end point} \\ t_f, & \text{final time: unknown} \end{cases}$$

Along with the Bolza formulation of Eqn. A-2, the following principle of optimality is applied (Girardet, 2014, Eqn. 2.10):

$$J^*(x(t_k), t_k) = \min_{x(.),a(.)} \int_{t_k}^{t_{k+1}} g(t, x(.), a(.))dt + J^*(x(t_{k+1}, t_{k+1}) \tag{A-3}$$

The resulting Hamilton-Jacobi equation is as follows (Girardet, 2014, Eqn. 2.11):

$$\nabla_t J^*(t, x) + \min_{a \in \mathbb{A}}[g(x, a) + \nabla_x J^*(t, x).f(x, a)] = 0 \tag{A-4}$$

For the path planning problem, the above equation can be reduced to the following equation, where u is the optimal cost representing the minimum arrival time (Girardet, 2014, Eqn. 2.12):

$$\max_{a \in \mathbb{A}}[-\nabla_x u(x).f(x, a)] = 1 \tag{A-5}$$

Eqn. A-5 can be solved using the Ordered Upwind algorithm, which offers a clever alternative to otherwise costly numerical methods for solving this problem. The Ordered Upwind algorithm can be considered as an adaptation of the Fast Marching Method, whose basic principle is highlighted in Figure A-4. Instead of tracking a moving interface (depicted in blue in Figure A-4), the interface is treated as still and nodes are systematically (in a similar way to Dijkstra's algorithm) updated with "crossing time" information (Sethian, 2010).

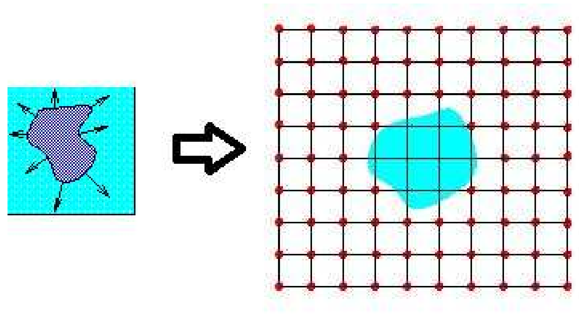


Figure A-4: Illustrations from Sethian's description of Fast Marching methods (Sethian, 2010)

In the case of the wind-optimal trajectory problem, however, the speed of the aircraft depends on direction; therefore, the speed is not considered isotropic and the Fast Marching methods are not applicable. As previously mentioned, the Ordered Upwind method is based on the same principles of the Fast Marching methods but has been adapted to take into account the “anisotropy ratio” (i.e. the ratio of fastest and slowest speed at a node) to compute the next unknown node information (Sethian, 2010). To better understand the ordered, Dijkstra-like movement between nodes, a schematic of the problem along with a description of the steps in calculating the value function are provided in Appendix B. It is interesting to note that the solution is developed by working “backwards”; therefore, in Girardet’s wind-optimal route problem, the propagating front emanates from the end point until the start point is reached and then the optimal path is reconstructed using the proper order from the start point to the end point (Girardet, 2014).

Hybrid Algorithm using Simulated Annealing

The overall approach taken by Girardet is to first optimize a single trajectory and then to consider all trajectories in order to make adjustments for de-congestion. A similar approach is taken by Ruiz and Solar, as shown by the simulation architecture in Figure A-5 (Ruiz & Soler, 2015). Though the particular methods used by Girardet differ from those used by Ruiz and Solar, a division into two scales similar to the “microscale” and “mesoscale” is used.

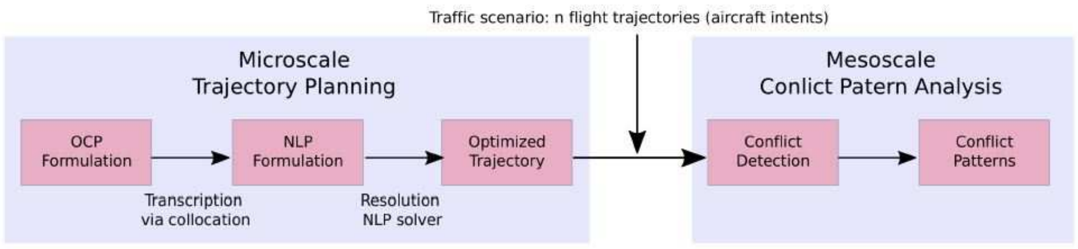

Traffic scenario: n flight trajectories (aircraft intents)
Microscale Trajectory Planning
OCP Formulation
Transcription via collocation
NLP Formulation
Resolution NLP solver
Optimized Trajectory
Mesoscale Conlict Patern Analysis
Conflict Detection
Conflict Patterns

Figure A-5: Simulation architecture divided into two scales: microscale and mesoscale (Ruiz & Soler, 2015, pp. 2)

For the optimization of a single trajectory (i.e. microscale), Girardet uses the Ordered Upwind method, as described in the previous section. For the treatment of the mesoscale, Girardet uses simulated annealing in an effort to clear areas of congestion (a model of congestion has been developed in the dissertation). The overall approach is considered a hybrid algorithm because the simulated annealing method takes the optimized microscale trajectory as input. A main reason for implementing the stochastic search method is the problem size; for example, there can be 500 trajectories to optimize on a given flight level (Girardet, 2014, pp. 130). As presented in Section A-2-3, metaheuristic methods do have the great benefit of a computational time that is independent from the problem complexity. However, it is suggested as future work in Girardet’s dissertation that computation time could be improved. Two additional suggestions for future work include the further refinement of simulated annealing parameters and ultimately an investigation of an optimization method other than simulated annealing (Girardet, 2014, pp. 150–151). It has indeed been referenced in Section A-2-3 that the parameters of the stochastic methods have a significant effect on the algorithm performance. In light of these suggestions as well as other practical concerns, such as the element of randomness in the search presenting potential complications during eventual certification processes, an alternative optimization method will be explored in the research proposed in the following chapter.

A-2-5 Summary

This chapter has offered a review of algorithms relevant to aircraft trajectory planning, their associated advantages and disadvantages, and their viability in terms of real life computational and operational applications. The following is a brief summary of the review along with lessons gleaned from conducting the review.

The literature related to UAV path planning situates the topic, in terms of referencing conventional methods and describing improvements upon those methods. In the work by researchers from Zhejiang University, for example, an important distinction is made within the general category of conventional methods: mathematical and graph-search methods (Yang et al., 2016). While the contribution of the proposed bio-inspired method is mostly beneficial to the specific problem of handling multiple UAVs, the organizing of conventional methods into the two categories helps to position the potential 4DT planner algorithms that appear later in the review. The remaining two studies treated in the UAV application group are similarly involved with multiple UAVs, but the 4PL method stands out among other mathematical methods as a potential candidate for 4DT planning (Upadhyay & Ratnoo, 2016). A graph-search, however, is less computationally complex and is therefore a primary candidate for the 4DT planner. The graph-search methods are further reviewed in the literature related to shortest path algorithms but it is already established that a similar characteristic among the conventional graph-search methods is a computational complexity that increases with the number of nodes in the graph (Sathyaraj et al., 2008). Also from the Sathyaraj reference, it is reported that the AStar algorithm out-performs the other three algorithms to which it is compared. Importantly, however, it should be noted that the heuristic upon which the AStar algorithm is based must be admissible, i.e. always under-estimate the remaining distance, in order to guarantee the shortest path.

With regard to shortest path algorithms, a time complexity comparison between Dijkstra's algorithm, Bellman Ford's algorithm and Floyd-Warshall's algorithm is provided (Baras & Theodorakopoulos, 2010). The comparison reveals that Dijkstra's algorithm has the least valued time complexity; however, the algorithm is limited by the constraint that all edges must have non-negative weights. It is generally understood that an algorithm will require less time if it visits less nodes; in this light, the branch-and-bound principle is introduced along with two search methods, the best-first and depth-first searches (Karlof, 2006).

Metaheuristic methods, namely the genetic algorithm and simulated annealing methods, are considered. These stochastic search methods offer the benefit of a runtime that is largely independent of the problem complexity (Rios & Lohn, 2009). Therefore, they can be used to handle the uncertainties that are inherent in a dynamic environment (Uçan & Altilar, 2012). However, runtime is an issue and the speed of convergence largely depends on the precision with which the algorithm parameters are set.

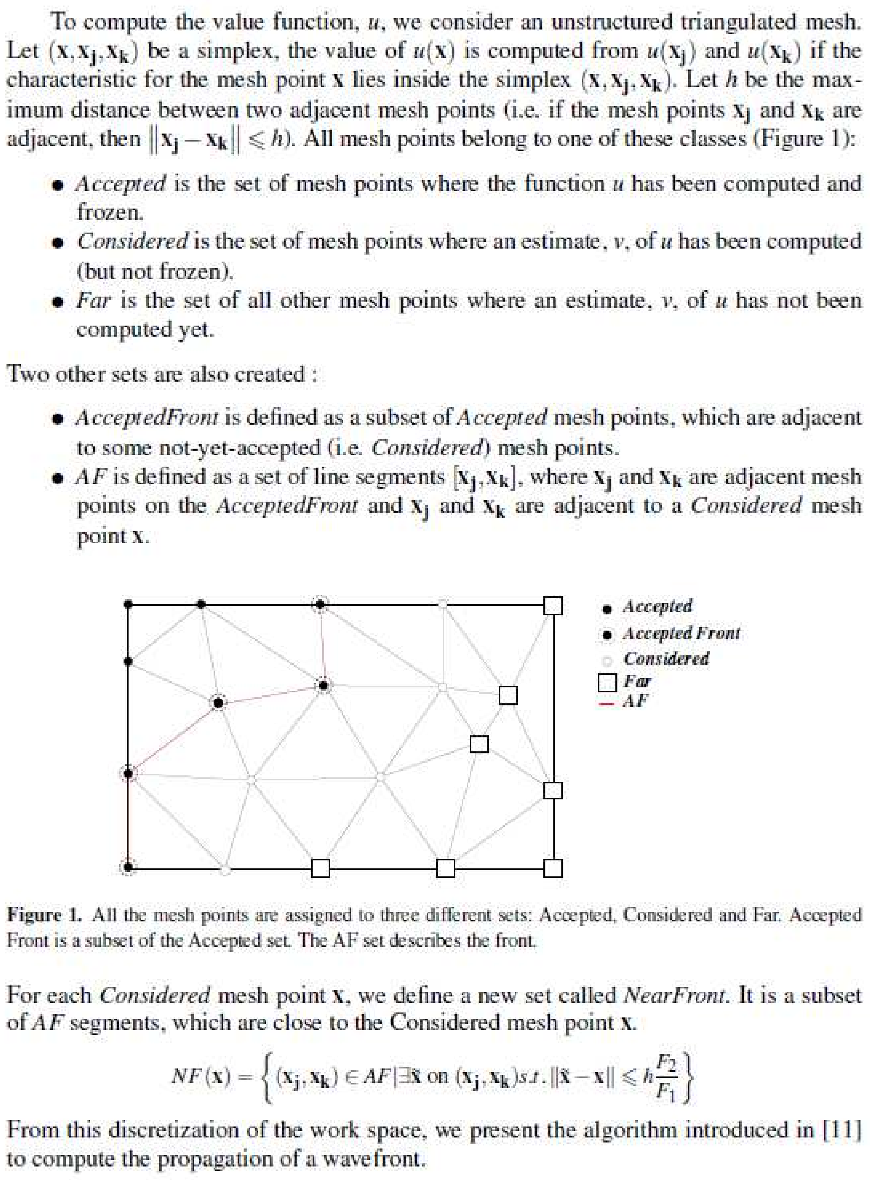
The work of Girardet is given special attention in the review (Girardet, 2014). The theory behind the Ordered Upwind method is discussed, followed by further details of the algorithm. Girardet's method of creating a wind-optimal trajectory using the Ordered Upwind algorithm and then using simulated annealing for the larger-scale problem to converge on a low-congestion (according to the congestion model developed in the dissertation) solution is analyzed. The suggestions for future work in Girardet's dissertation offer guidance for the approach to use in developing the 4DT planner.

The 4DT planner can be inspired by a combination of algorithms detailed in this review. Namely, the Ordered Upwind method presented by Girardet can be used to create a wind-optimal route; however, this wind-optimal route should be adjusted using a method other than simulated annealing. Minimal adjustments to the wind-optimal route can be made in order to avoid all obstacles; these deviations around obstacles should be calculated using a branching method that minimizes the user-defined cost. This determination is motivated by the insight that any obstacle can be modeled as a convex polygon (Upadhyay & Ratnoo, 2016). Combined spatial maneuvering above, below, to the left or right of all presented obstacles are therefore the options to consider in a heuristic branching method.

Appendix B

Ordered Upwind Algorithm

A definition of the Ordered Upwind algorithm is included in the following two pages. The pages are excerpts, imported in the form of figures, from Girardet's 2013 publication, "Generating optimal aircraft trajectories with respect to weather conditions" (Girardet et al., 2013). Figure B-1 defines the types of nodes in the considered mesh. Figure B-2 lists the steps of the algorithm itself.



To compute the value function, u, we consider an unstructured triangulated mesh. Let $(\mathbf{x},\mathbf{x_j},\mathbf{x_k})$ be a simplex, the value of $u(\mathbf{x})$ is computed from $u(\mathbf{x_j})$ and $u(\mathbf{x_k})$ if the characteristic for the mesh point $\mathbf{x}$ lies inside the simplex $(\mathbf{x},\mathbf{x_j},\mathbf{x_k})$. Let h be the maximum distance between two adjacent mesh points (i.e. if the mesh points $\mathbf{x_j}$ and $\mathbf{x_k}$ are adjacent, then $\|\mathbf{x_j}-\mathbf{x_k}\| \leqslant h$). All mesh points belong to one of these classes (Figure 1):

- *Accepted* is the set of mesh points where the function u has been computed and frozen.
- *Considered* is the set of mesh points where an estimate, v, of u has been computed (but not frozen).
- *Far* is the set of all other mesh points where an estimate, v, of u has not been computed yet.

Two other sets are also created :

- *AcceptedFront* is defined as a subset of *Accepted* mesh points, which are adjacent to some not-yet-accepted (i.e. *Considered*) mesh points.
- AF is defined as a set of line segments $[\mathbf{x_j},\mathbf{x_k}]$, where $\mathbf{x_j}$ and $\mathbf{x_k}$ are adjacent mesh points on the *AcceptedFront* and $\mathbf{x_j}$ and $\mathbf{x_k}$ are adjacent to a *Considered* mesh point $\mathbf{x}$.

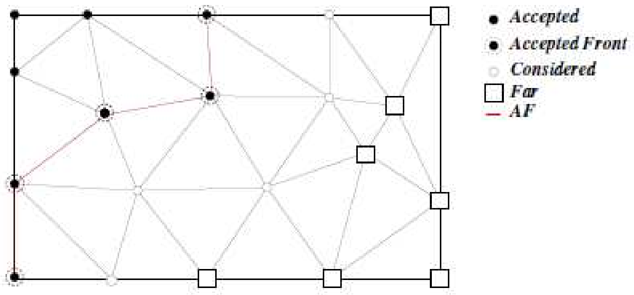


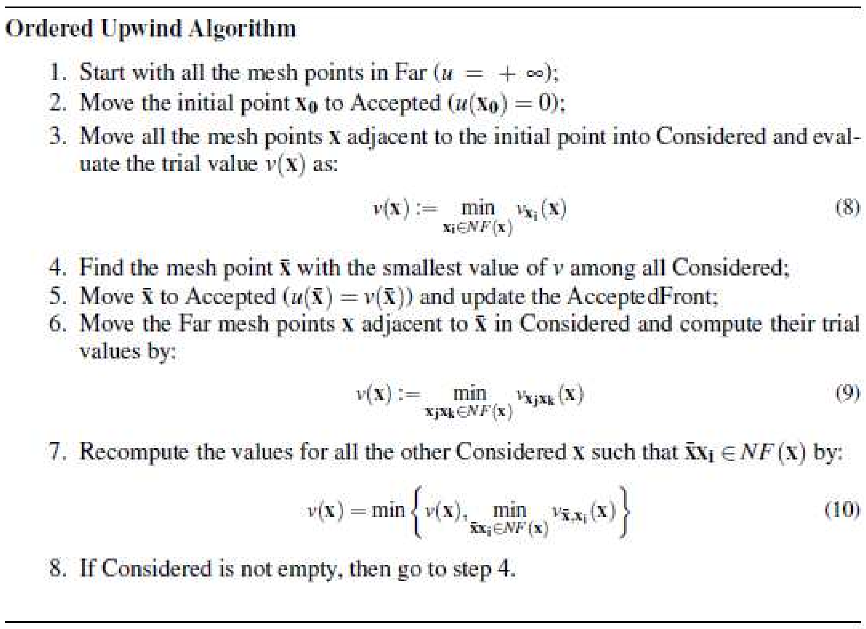
Figure 1. All the mesh points are assigned to three different sets: Accepted, Considered and Far. Accepted Front is a subset of the Accepted set. The AF set describes the front.

For each *Considered* mesh point $\mathbf{x}$, we define a new set called *NearFront*. It is a subset of AF segments, which are close to the Considered mesh point $\mathbf{x}$.

$$NF(\mathbf{x}) = \left\{ (\mathbf{x_j},\mathbf{x_k}) \in AF \mid \exists \tilde{\mathbf{x}} \text{ on } (\mathbf{x_j},\mathbf{x_k}) s.t. \|\tilde{\mathbf{x}} - \mathbf{x}\| \leqslant h\frac{F_2}{F_1} \right\}$$

From this discretization of the work space, we present the algorithm introduced in [11] to compute the propagation of a wavefront.

Figure B-1: Excerpt to show types of nodes in mesh (Girardet et al., 2013, pp. 7)



Ordered Upwind Algorithm

1. Start with all the mesh points in Far ($u = +\infty$);
2. Move the initial point $\mathbf{x_0}$ to Accepted ($u(\mathbf{x_0}) = 0$);
3. Move all the mesh points $\mathbf{x}$ adjacent to the initial point into Considered and evaluate the trial value $v(\mathbf{x})$ as:

$$v(\mathbf{x}) := \min_{\mathbf{x_i} \in NF(\mathbf{x})} v_{\mathbf{x_i}}(\mathbf{x}) \tag{8}$$

4. Find the mesh point $\bar{\mathbf{x}}$ with the smallest value of v among all Considered;
5. Move $\bar{\mathbf{x}}$ to Accepted ($u(\bar{\mathbf{x}}) = v(\bar{\mathbf{x}})$) and update the AcceptedFront;
6. Move the Far mesh points $\mathbf{x}$ adjacent to $\bar{\mathbf{x}}$ in Considered and compute their trial values by:

$$v(\mathbf{x}) := \min_{\mathbf{x_j x_k} \in NF(\mathbf{x})} v_{\mathbf{x_j x_k}}(\mathbf{x}) \tag{9}$$

7. Recompute the values for all the other Considered $\mathbf{x}$ such that $\bar{\mathbf{x}}\mathbf{x_i} \in NF(\mathbf{x})$ by:

$$v(\mathbf{x}) = \min\left\{ v(\mathbf{x}), \min_{\bar{\mathbf{x}}\mathbf{x_i} \in NF(\mathbf{x})} v_{\bar{\mathbf{x}},\mathbf{x_i}}(\mathbf{x}) \right\} \tag{10}$$

8. If Considered is not empty, then go to step 4.

Figure B-2: Excerpt to show steps in Ordered Upwind algorithm (Girardet et al., 2013, pp. 7–8)

Appendix C

Python Coding Workflow

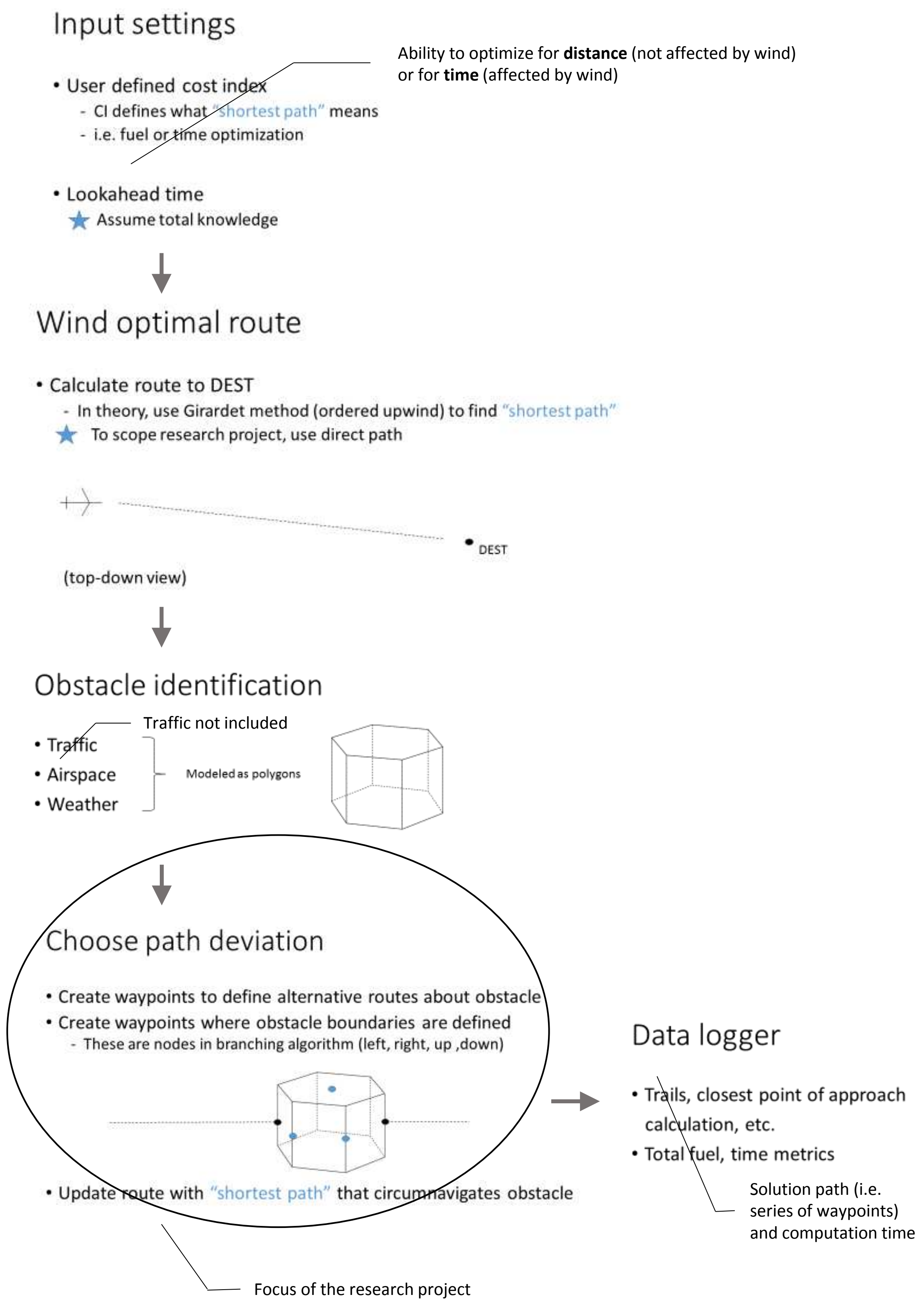
C-1 Evolution of Coding Design

The evolution of the heuristic branching planner's coding workflow is recorded in this section. These details are included to offer the reader a better understanding of the original design concept, as well as the assumptions that have been made for the purpose of scoping the research project. Suggestions to future researchers are informed by this information, and will be further explained in Appendix E.

Figure C-1 is a reproduction of the code flow envisioned at the time of the preliminary thesis report with annotations to indicate the final design. The "Choose path deviation" block in Figure C-1 is circled to emphasize the focus of this research project. Rather than "creating" waypoints to define alternative routes about an obstacle, the obstacle vertices themselves are used in the final design. It should also be noted that in the final design, branching decisions are not made at the level of the individual obstacle, as may be suggested by Figure C-1, but are made according to the algorithm described in Part I.

The final design of the heuristic branching planner has been implemented in two Python modules: "PathPlanningFINAL" and "mytoolsFINAL". The planning module contains the algorithm described through pseudocode in Part I (reproduced in Figure C-2), and the tools module contains necessary supporting class and function definitions. Figure C-2 offers a schematic of the two modules that make up the heurisitic branching planner. The sample code used in Figure C-2 to represent the contents of the tools module is an actual windfield definition; the windfield definition is one of many definitions that can be found in this module and are essential to the overall trajectory planner.

Although the windfield is defined within the planner code itself, the obstacle field is defined in a text file that is used as input to the planner. Likewise, the origin and destination coordinates (followed by two values to represent altitude and true airspeed throughout the trajectory) are defined in a text file to be used as input.


Input settings

- User defined cost index
  - CI defines what "shortest path" means
  - i.e. fuel or time optimization

Ability to optimize for **distance** (not affected by wind) or for **time** (affected by wind)

- Lookahead time
  - ★ Assume total knowledge

Wind optimal route

- Calculate route to DEST
  - In theory, use Girardet method (ordered upwind) to find "shortest path"
  - ★ To scope research project, use direct path

Obstacle identification

- Traffic
- Airspace
- Weather

Traffic not included

Modeled as polygons

Choose path deviation

- Create waypoints to define alternative routes about obstacle
- Create waypoints where obstacle boundaries are defined
  - These are nodes in branching algorithm (left, right, up ,down)
- Update route with "shortest path" that circumnavigates obstacle

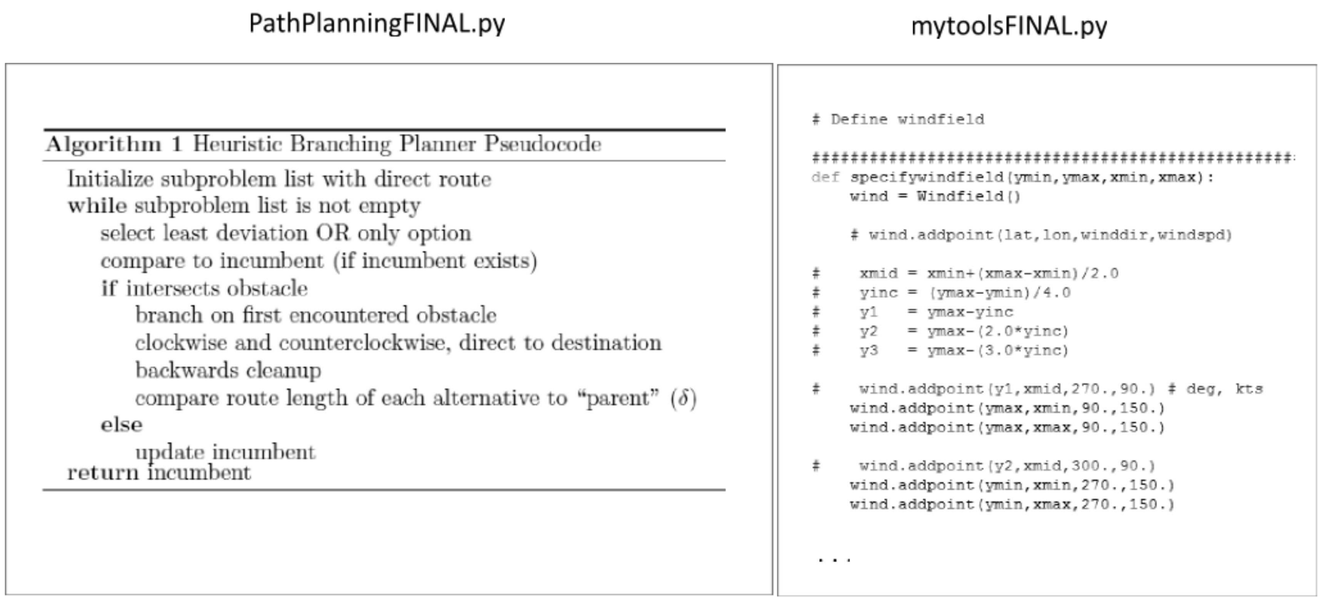
Focus of the research project

Data logger

- Trails, closest point of approach calculation, etc.
- Total fuel, time metrics

Solution path (i.e. series of waypoints) and computation time

Figure C-1: Trajectory Planner Coding Workflow



```
Algorithm 1 Heuristic Branching Planner Pseudocode
  Initialize subproblem list with direct route
  while subproblem list is not empty
      select least deviation OR only option
      compare to incumbent (if incumbent exists)
      if intersects obstacle
          branch on first encountered obstacle
          clockwise and counterclockwise, direct to destination
          backwards cleanup
          compare route length of each alternative to “parent” ($\delta$)
      else
          update incumbent
  return incumbent
```

```
# Define windfield

###################################################
def specifywindfield(ymin,ymax,xmin,xmax):
    wind = Windfield()

    # wind.addpoint(lat,lon,winddir,windspd)

#     xmid = xmin+(xmax-xmin)/2.0
#     yinc = (ymax-ymin)/4.0
#     y1   = ymax-yinc
#     y2   = ymax-(2.0*yinc)
#     y3   = ymax-(3.0*yinc)

#     wind.addpoint(y1,xmid,270.,90.) # deg, kts
    wind.addpoint(ymax,xmin,90.,150.)
    wind.addpoint(ymax,xmax,90.,150.)

#     wind.addpoint(y2,xmid,300.,90.)
    wind.addpoint(ymin,xmin,270.,150.)
    wind.addpoint(ymin,xmax,270.,150.)

 ...
```

Figure C-2: Trajectory Planner Modules with Representative Code

C-2 Planner Comparison: Experimental Setup

In order to evaluate the heuristic branching planner, an experiment folder is created with the necessary files and folders. The experiment folder contents are shown in Figure C-3 to illustrate the overall structure of the experiment.

As shown in Figure C-3, the Branching planner modules and Girardet planner modules can be found in their respective planner folders. The driver module that executes the experiment sequence can be found directly in the experiment folder; this module calls on the two planners in turn. Before calling the planners, however, the driver module reads from the text file labeled "inputfile" in Figure C-3 since it contains the names of the text files to use within the Obstacles and Routes folders. The appropriate files are then used as input to the planners, and the solutions output by the planners are saved in the Logs folder. The comparison and dependent measures modules are then called upon to make graphs of the results (to be saved in the Analysis folder) and to calculate defined metrics (to be saved in the Logs folder).

With this setup, many repetitions may be run without user input by simply defining a sequence of multiple filenames from the Obstacles and Routes folders within inputfile.txt before running the driver module. It is convenient that the inputs for the two planners are defined in one place, rather than having to make changes to files within the individual folders of the planners. It is important to note, however, that the windfield definition must still be manually changed within the code of each planner (and also in the code of the wind plotter module) before running the driver module. As noted in the technical paper of Part I, all plotting commands have been removed from the planners in the experiment folder in an effort to normalize runtime measures. The user-friendly selection of the input files has also been removed for the batch processing required by the experiment.

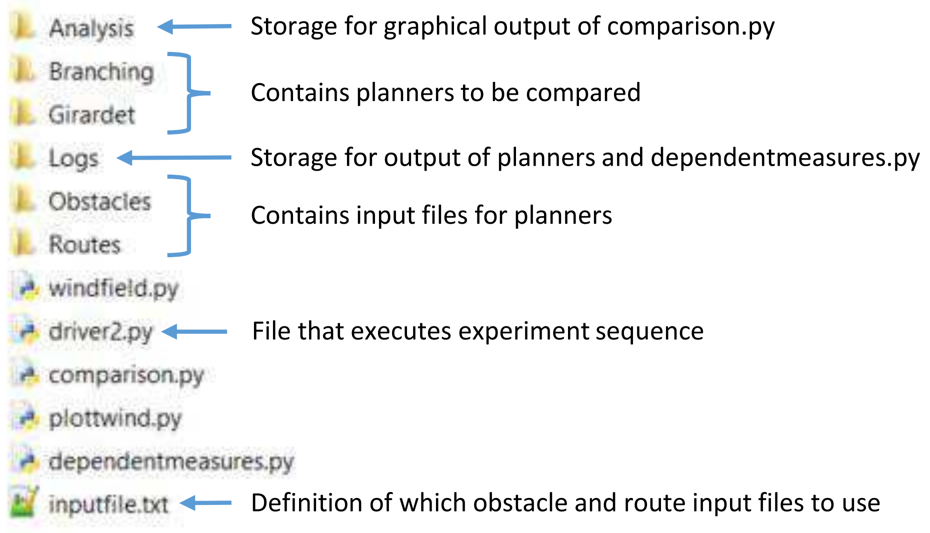

Analysis
Storage for graphical output of comparison.py
Branching
Girardet
Contains planners to be compared
Logs
Storage for output of planners and dependentmeasures.py
Obstacles
Routes
Contains input files for planners
windfield.py
driver2.py
File that executes experiment sequence
comparison.py
plottwind.py
dependentmeasures.py
inputfile.txt
Definition of which obstacle and route input files to use

Figure C-3: Contents of Experiment Folder

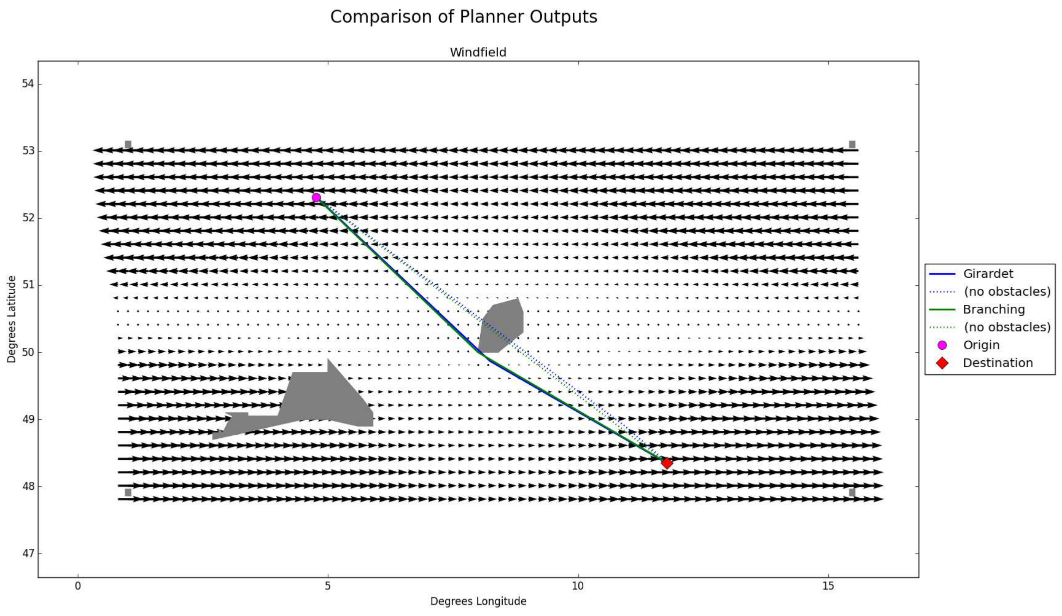
Appendix D

Graphical Results

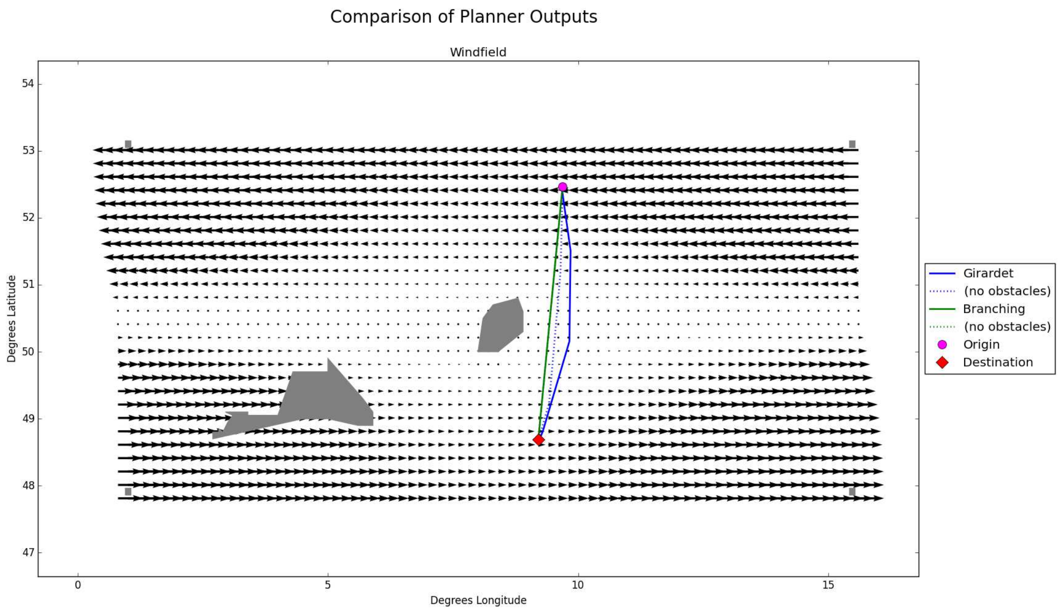
The graphical results for all experimental conditions are provided in this appendix.

Figures D-1 through D-9 represent the nine experimental conditions. All low windfield complexity conditions (ranging from low to high obstacle field complexity) are treated first in Figures D-1 through D-3. Medium windfield complexity conditions (also from low to high obstacle field complexity) are treated next in Figures D-4 through D-6. High windfield complexity conditions (again from low to high obstacle field complexity) are treated in Figures D-7 through D-9.

Each figure thus represents one experimental condition and contains parts (**a**) through (**h**) to represent the eight city pair repetitions.

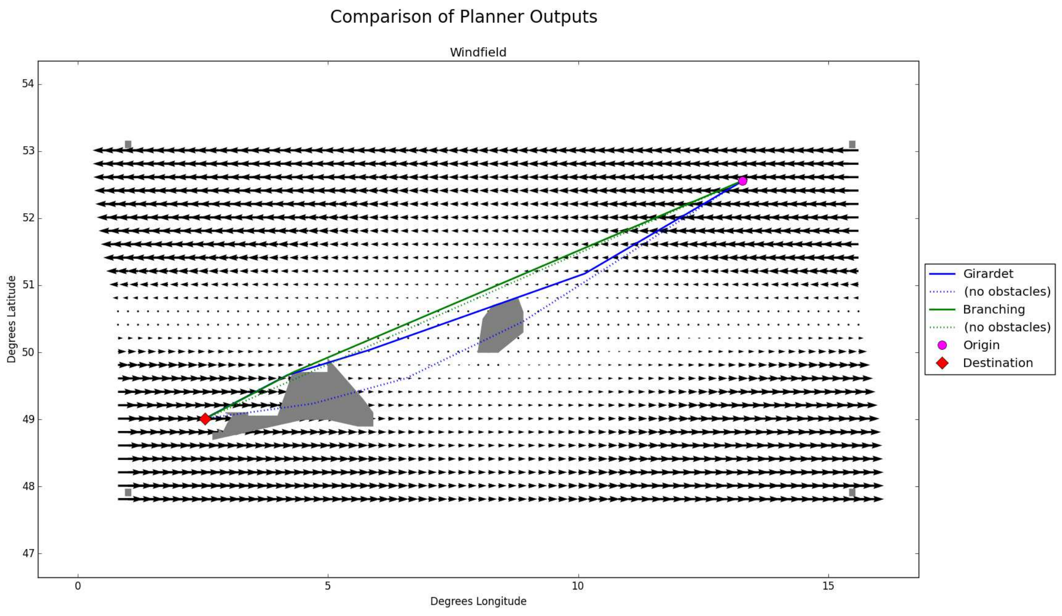


(a)

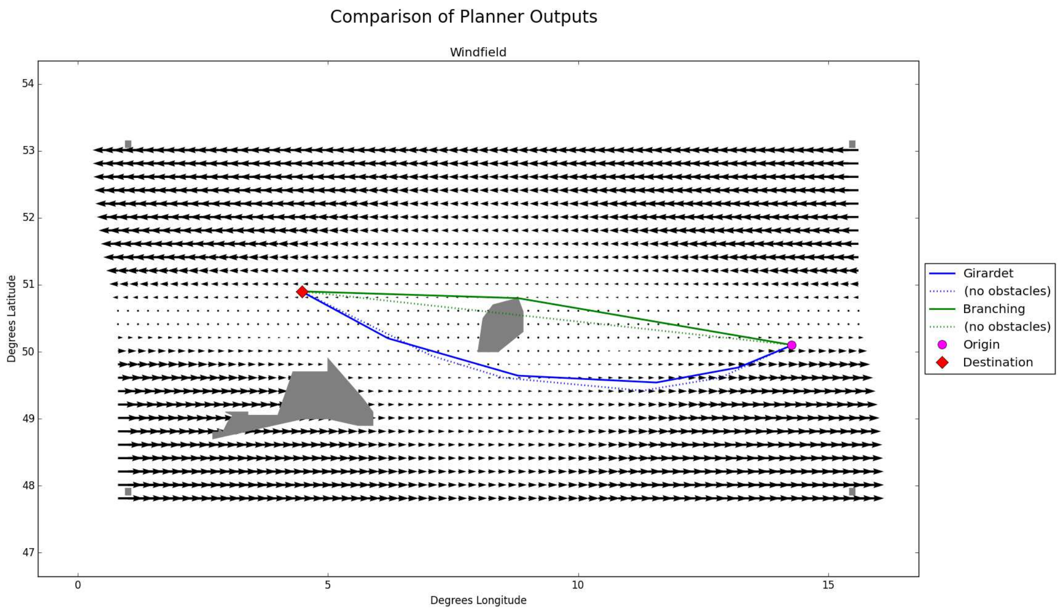


(b)

Figure D-1: Graphical Results for Low Windfield Complexity, Low Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

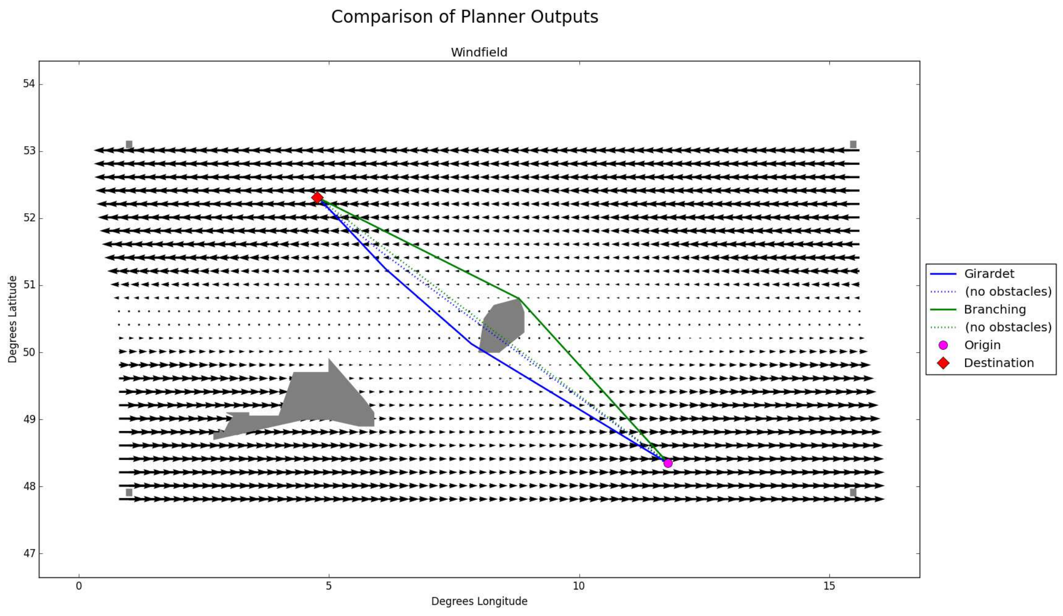


(c)

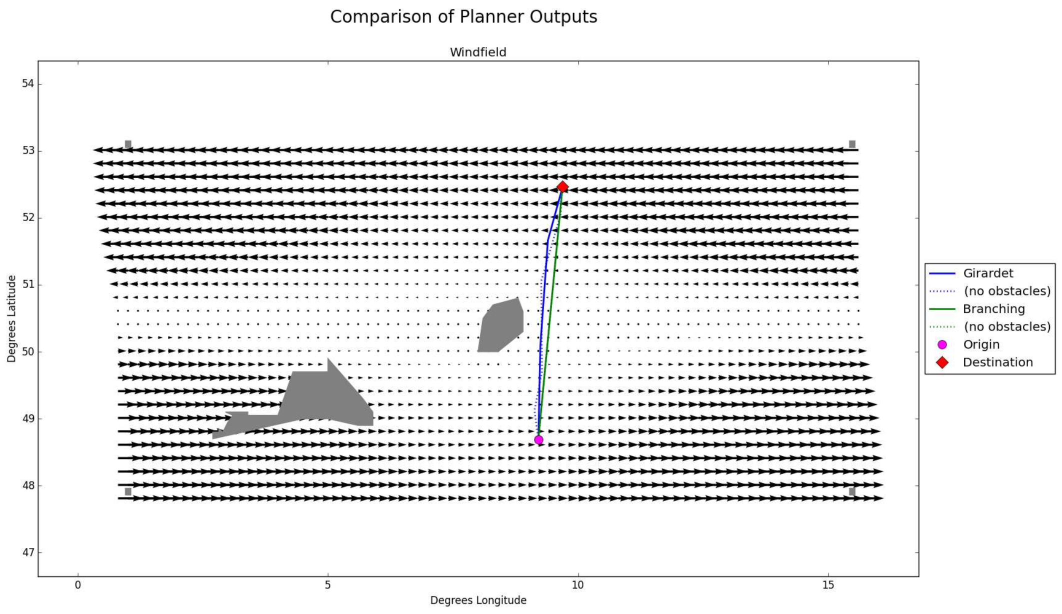


(d)

Figure D-1: (continued) Graphical Results for Low Windfield Complexity, Low Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

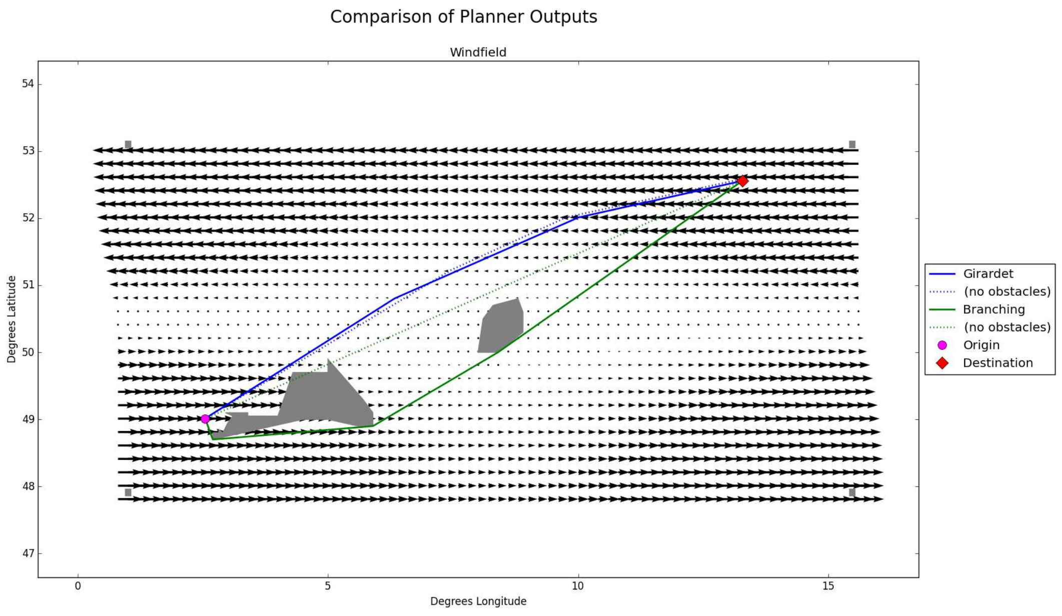


(e)

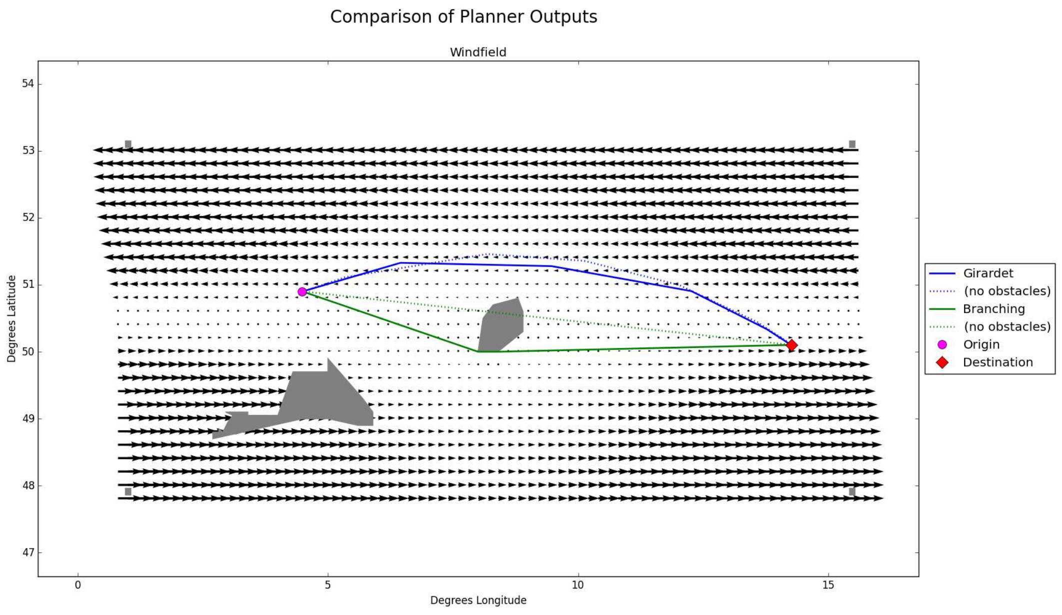


(f)

Figure D-1: (continued) Graphical Results for Low Windfield Complexity, Low Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

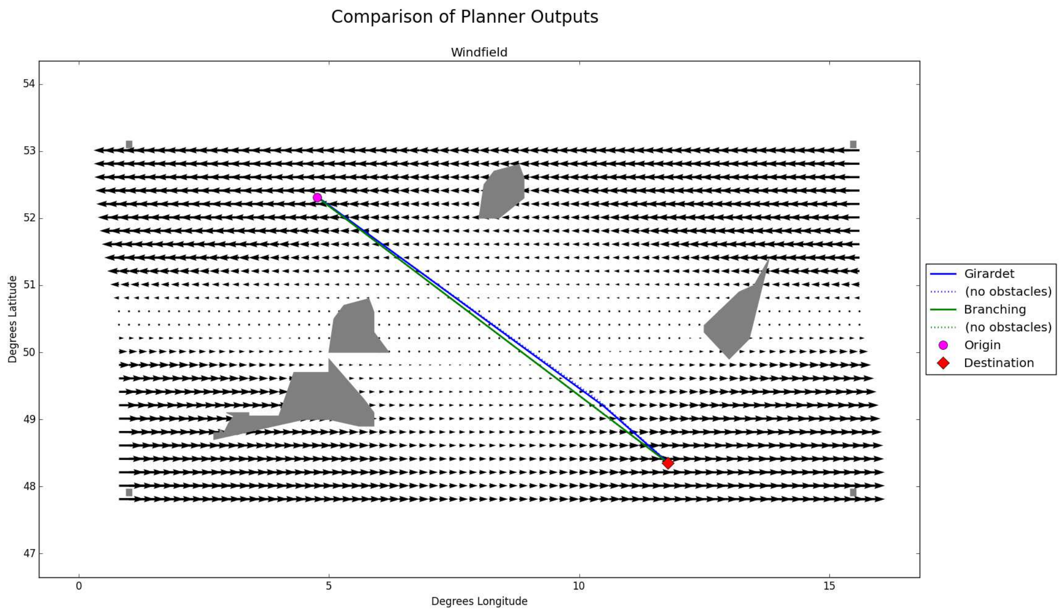


(g)

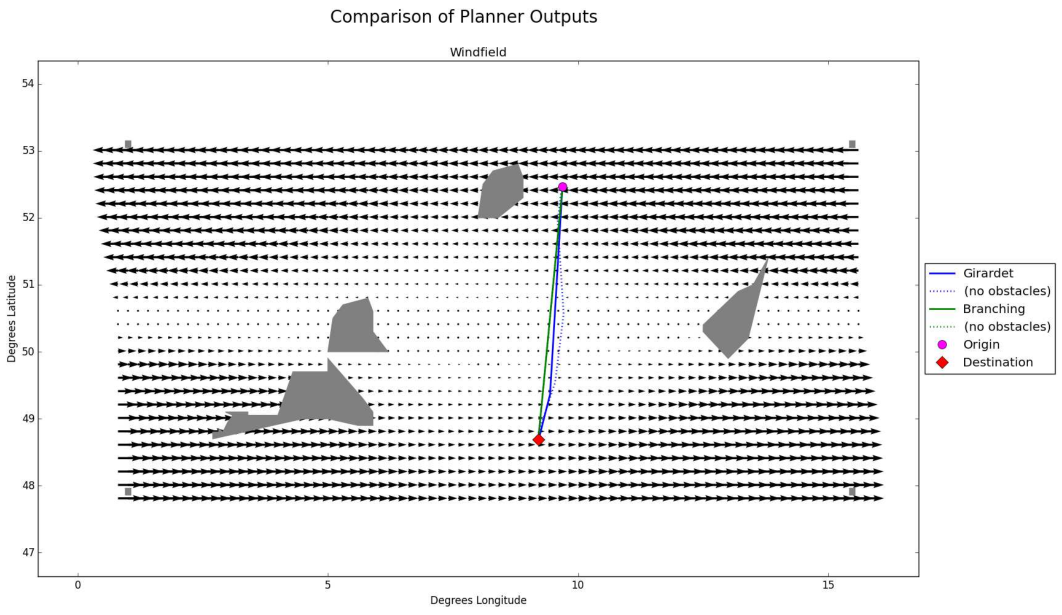


(h)

Figure D-1: (continued) Graphical Results for Low Windfield Complexity, Low Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

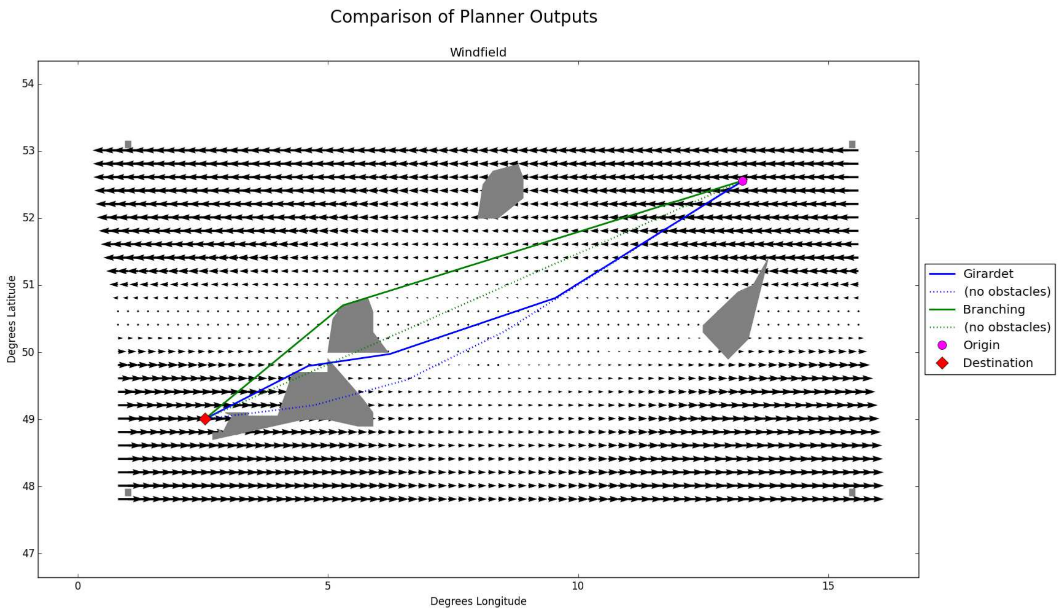


(a)

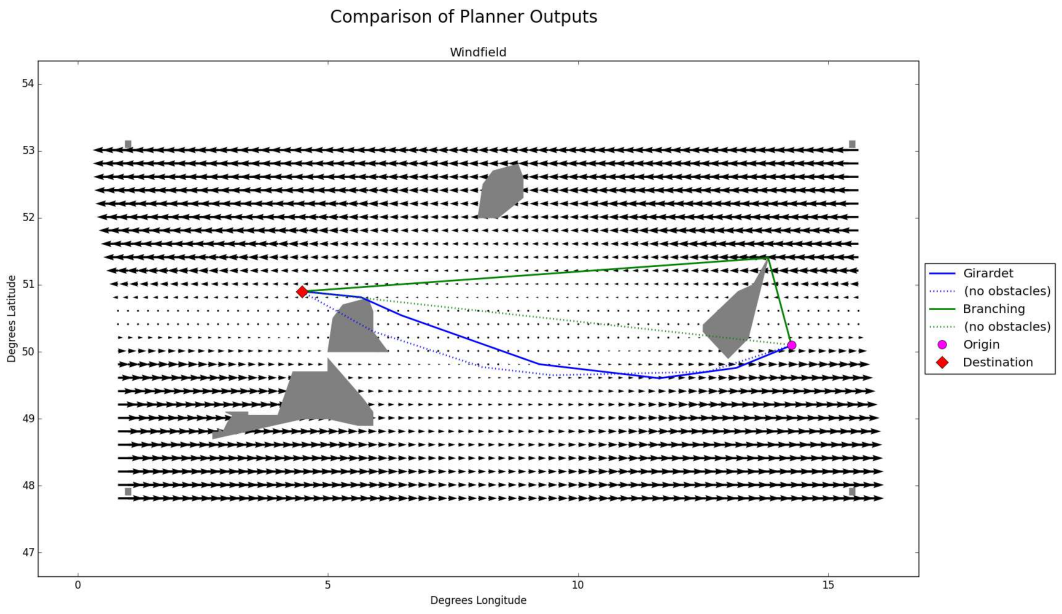


(b)

Figure D-2: Graphical Results for Low Windfield Complexity, Medium Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

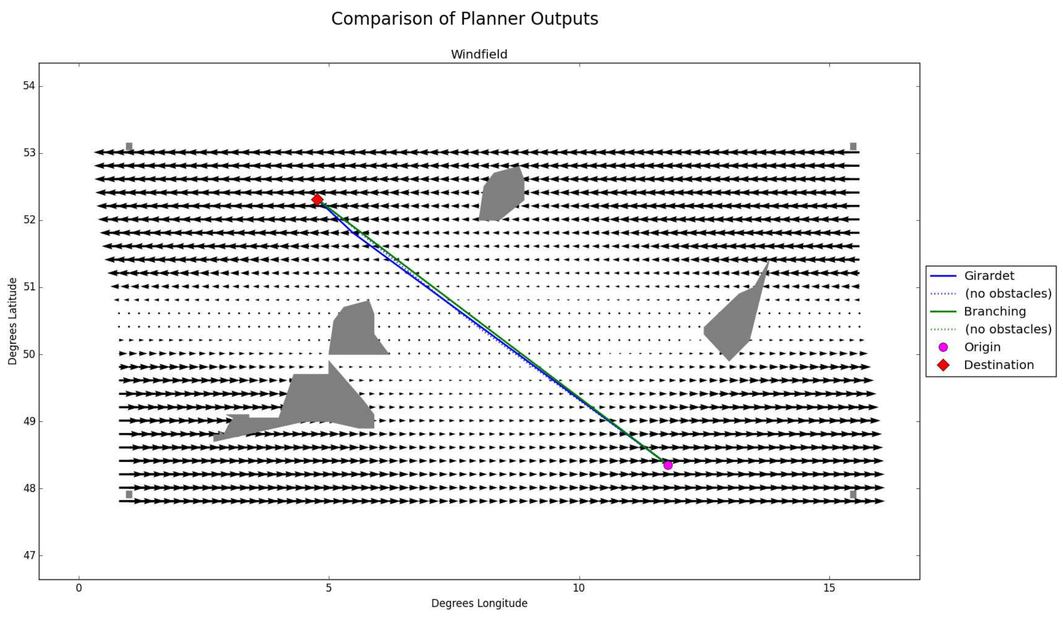


(c)

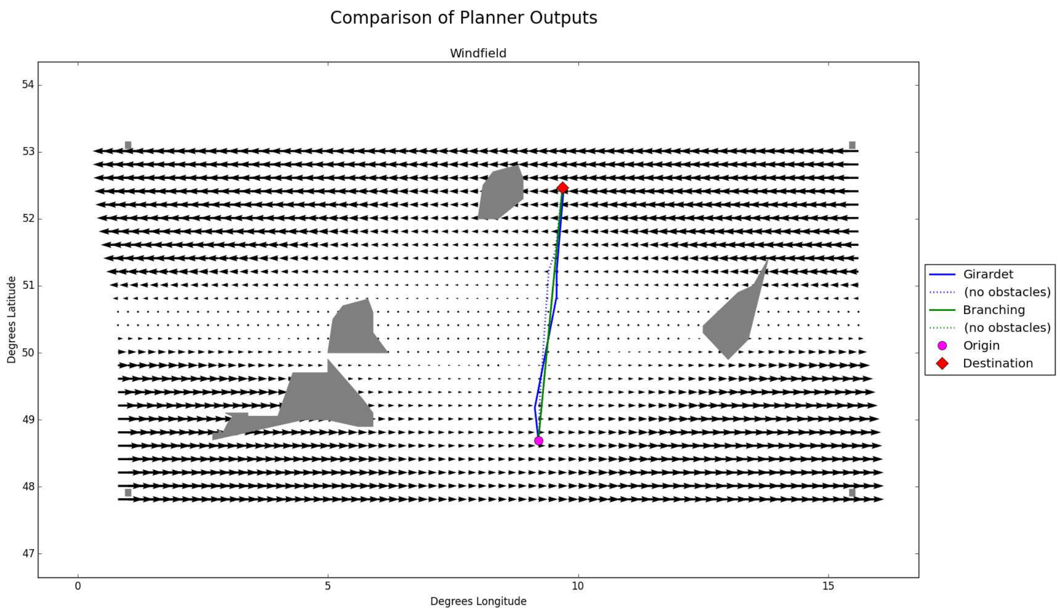


(d)

Figure D-2: (continued) Graphical Results for Low Windfield Complexity, Medium Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

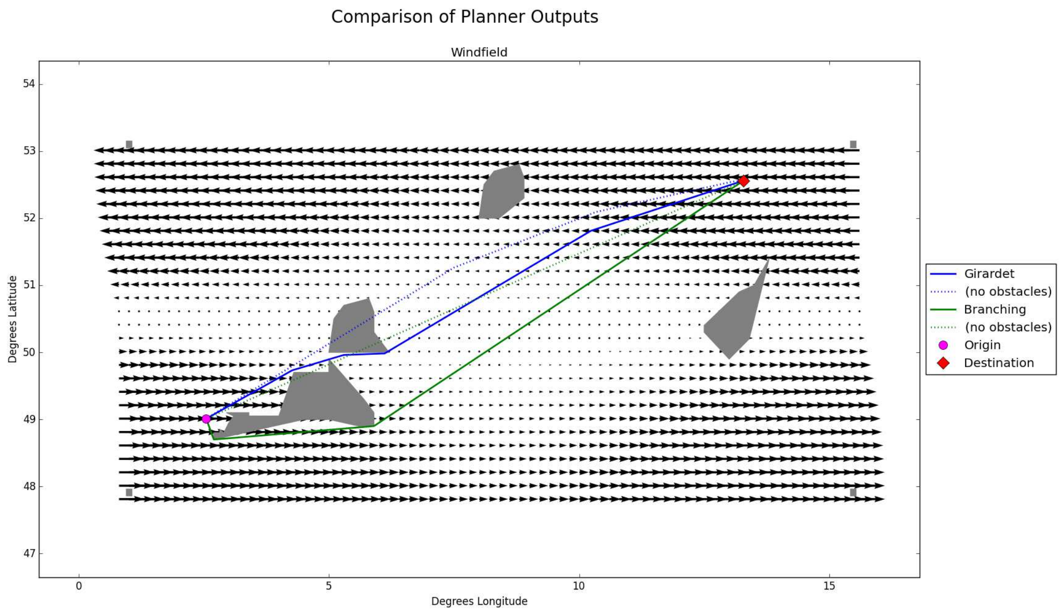


(e)

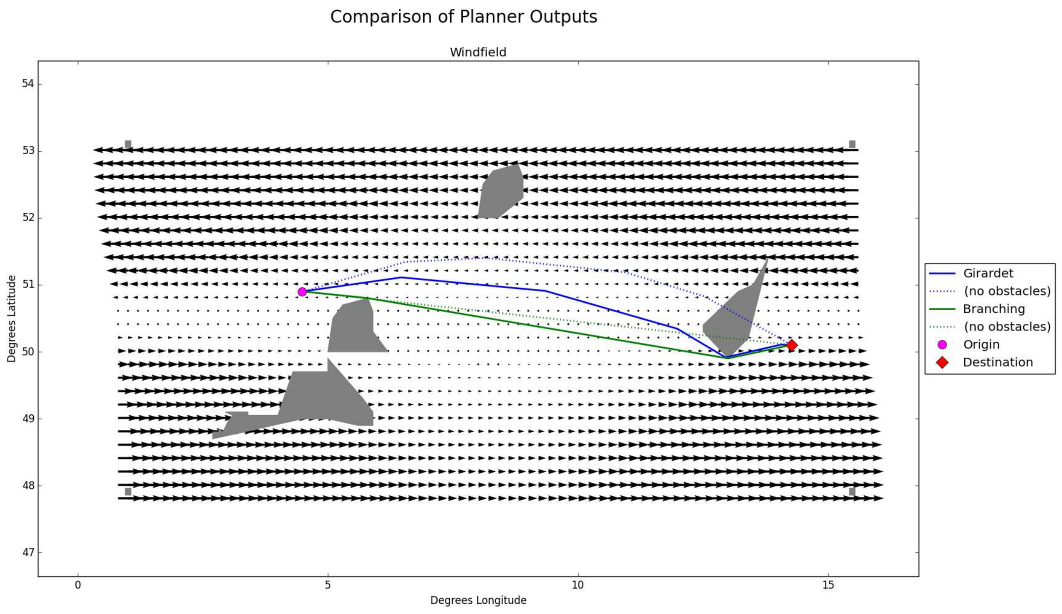


(f)

Figure D-2: (continued) Graphical Results for Low Windfield Complexity, Medium Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

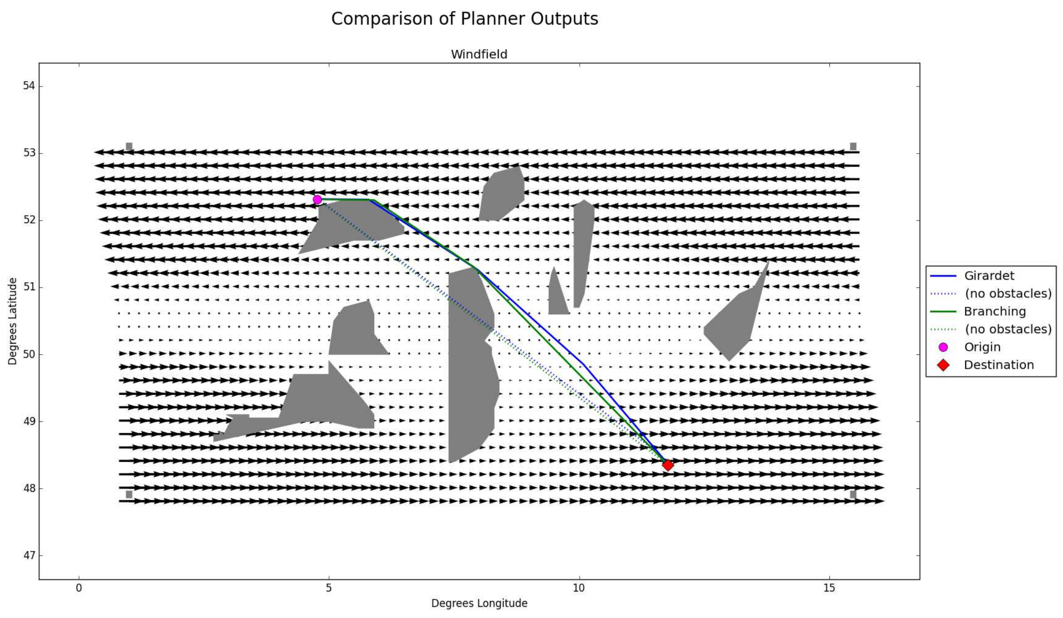


(g)

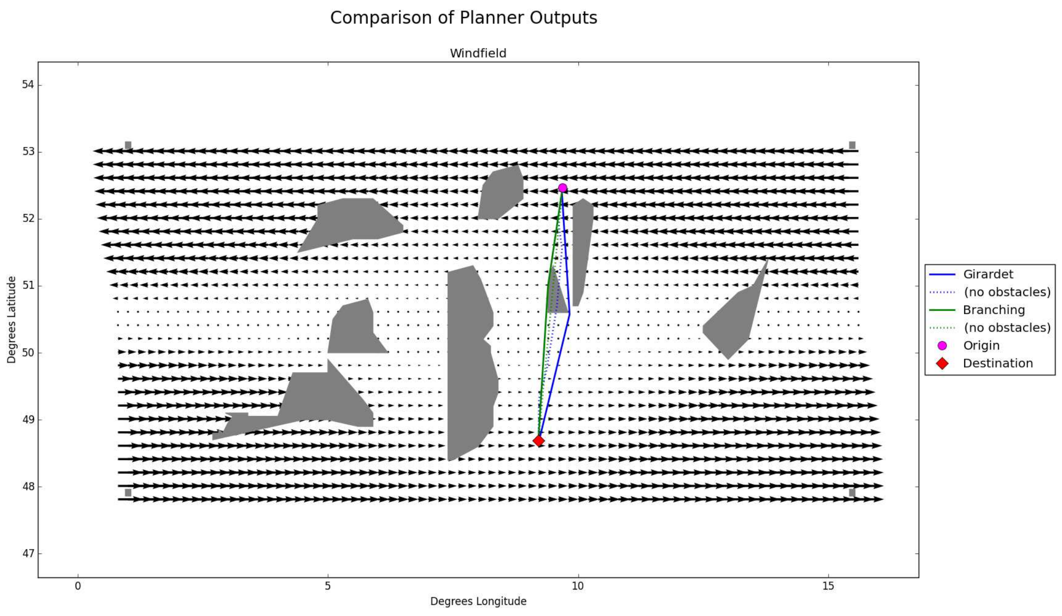


(h)

Figure D-2: (continued) Graphical Results for Low Windfield Complexity, Medium Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

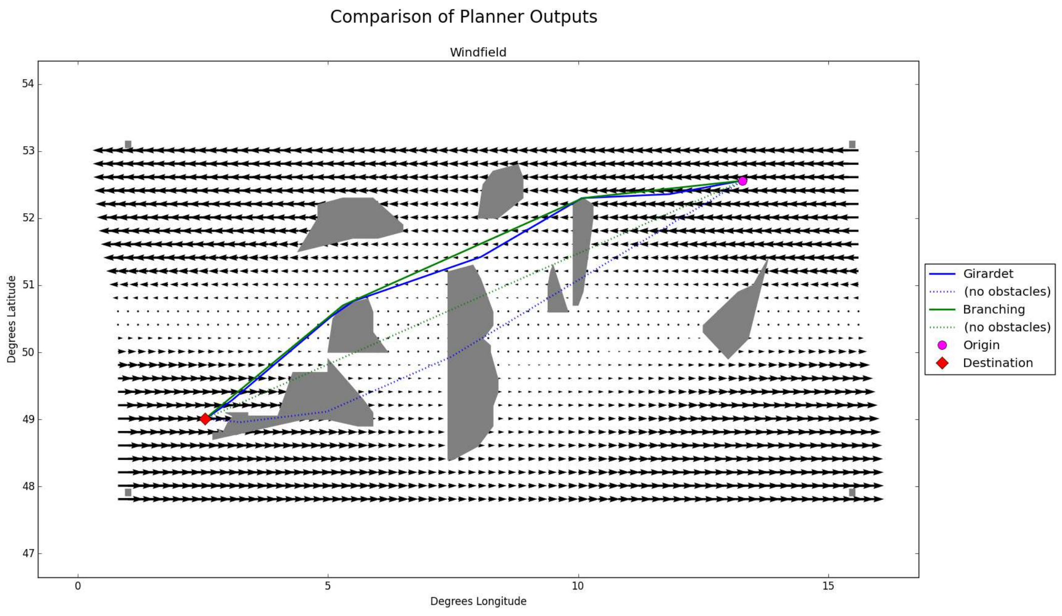


(a)

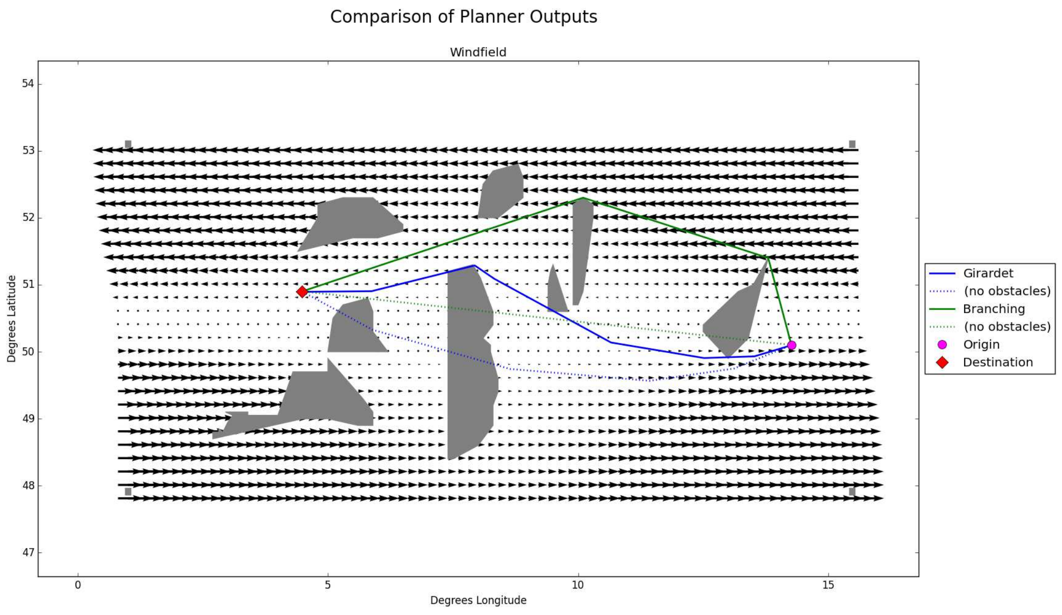


(b)

Figure D-3: Graphical Results for Low Windfield Complexity, High Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

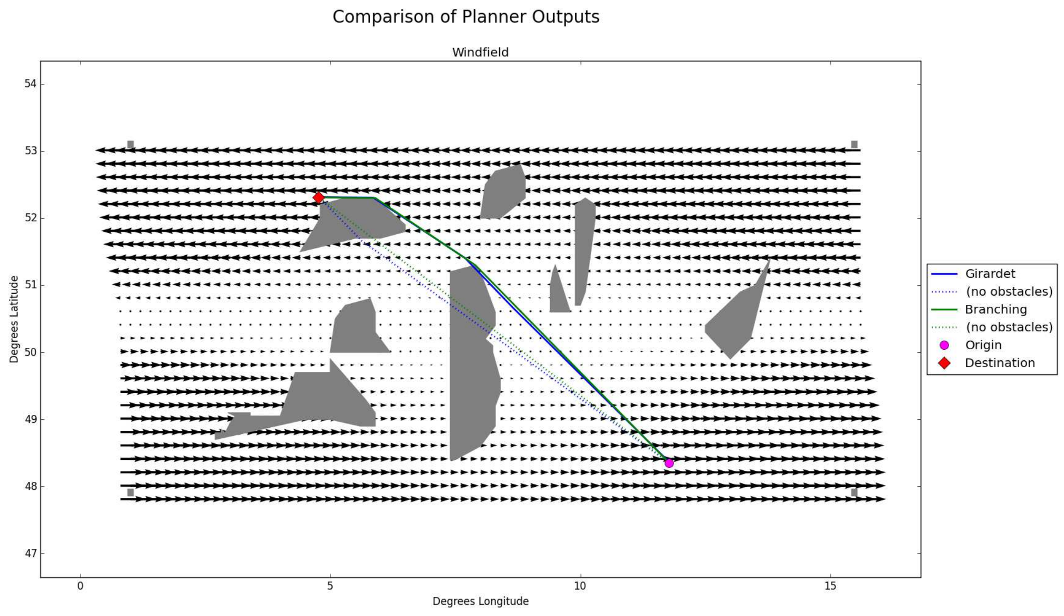


(c)

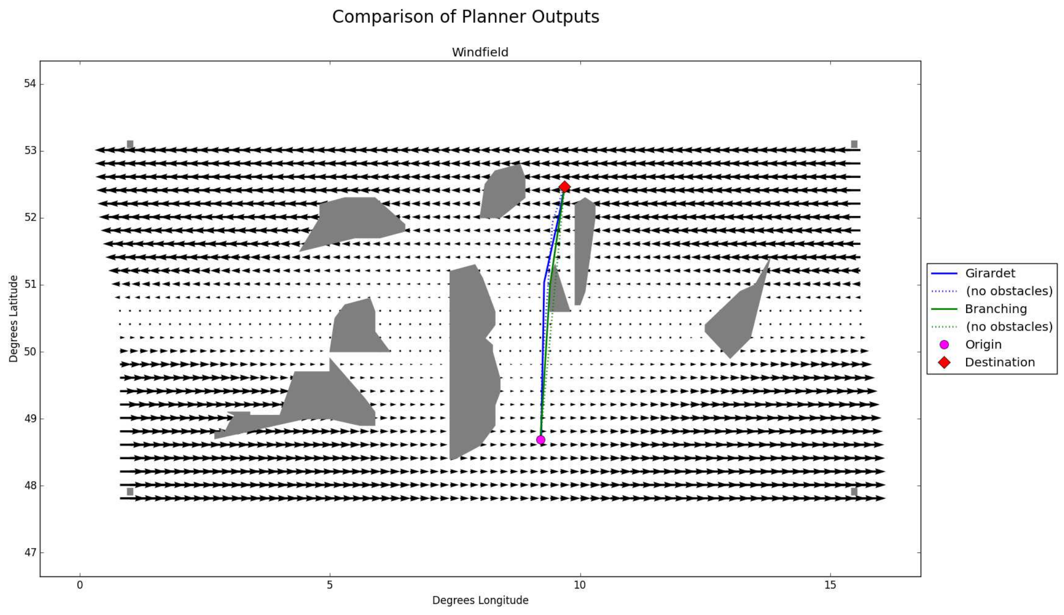


(d)

Figure D-3: (continued) Graphical Results for Low Windfield Complexity, High Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

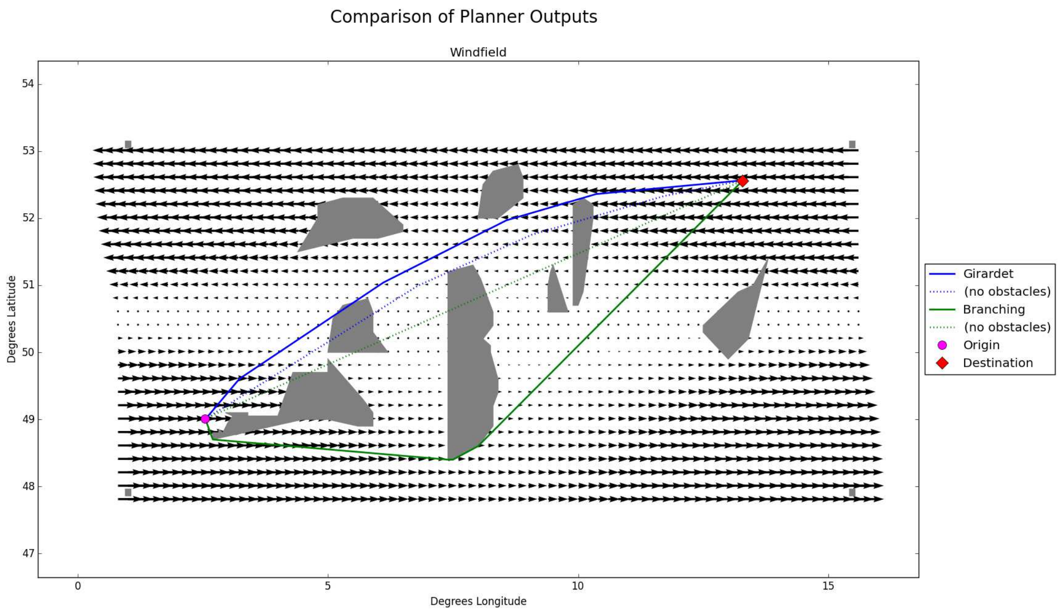


(e)

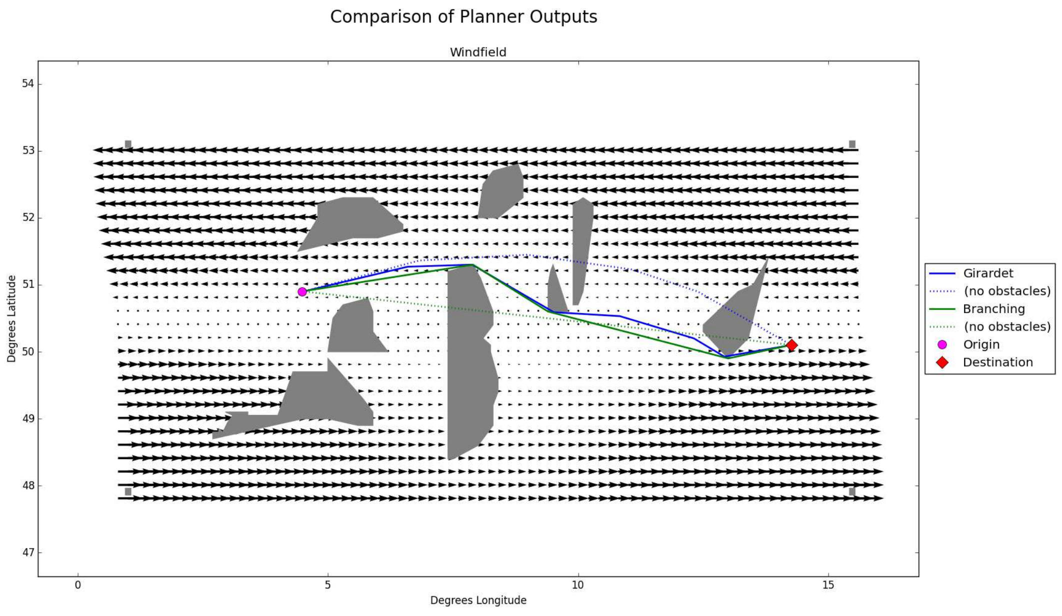


(f)

Figure D-3: (continued) Graphical Results for Low Windfield Complexity, High Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

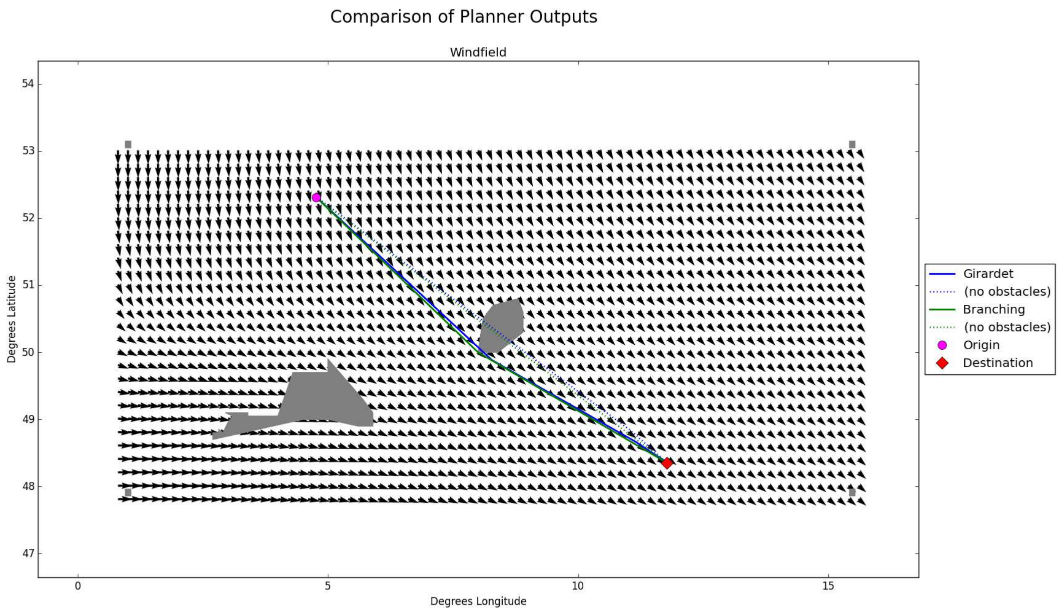


(g)

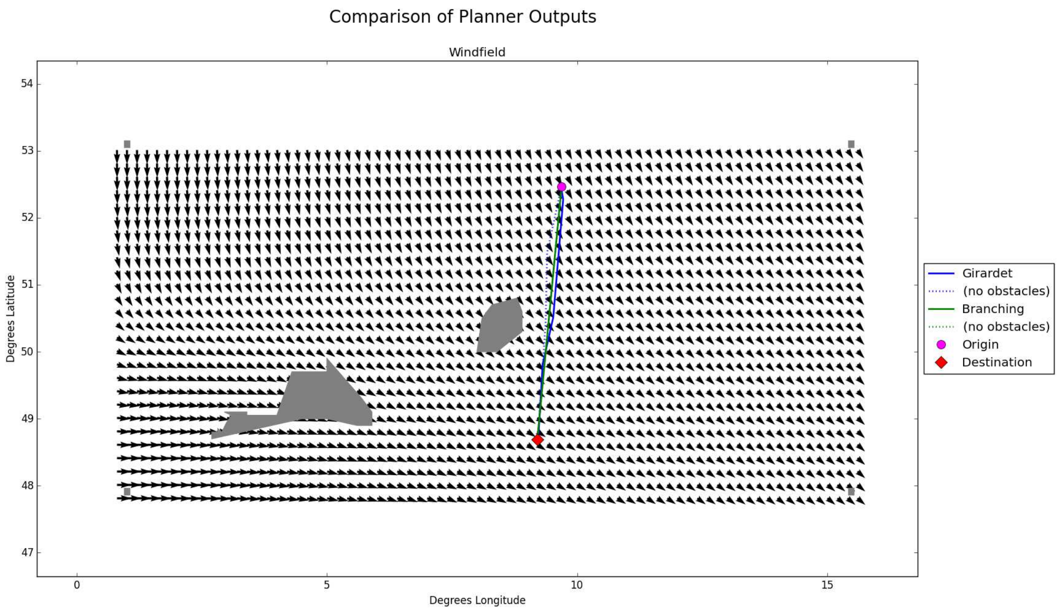


(h)

Figure D-3: (continued) Graphical Results for Low Windfield Complexity, High Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

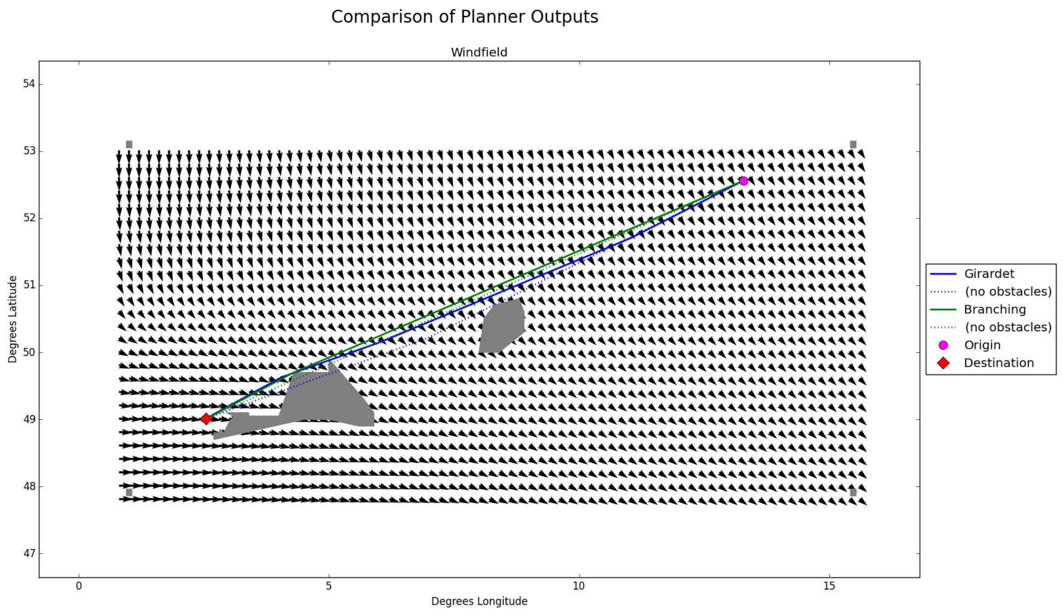


(a)

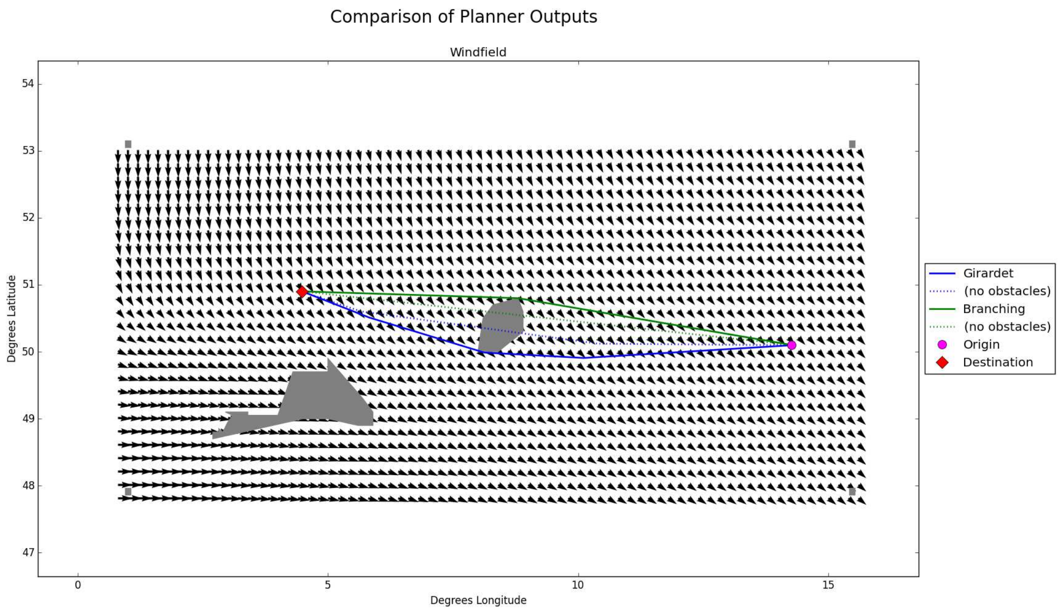


(b)

Figure D-4: Graphical Results for Medium Windfield Complexity, Low Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

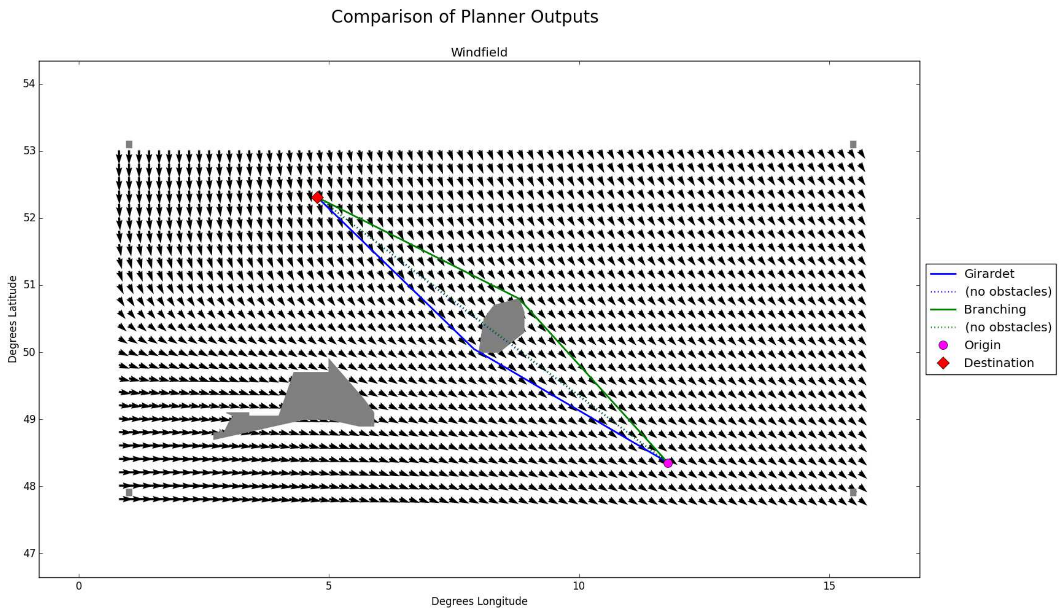


(c)

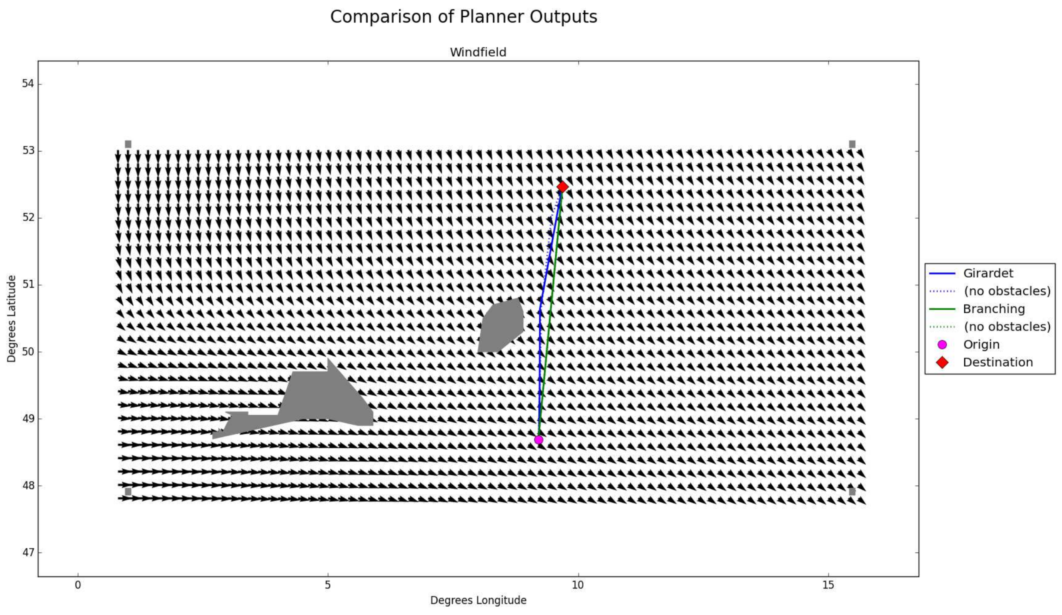


(d)

Figure D-4: (continued) Graphical Results for Medium Windfield Complexity, Low Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

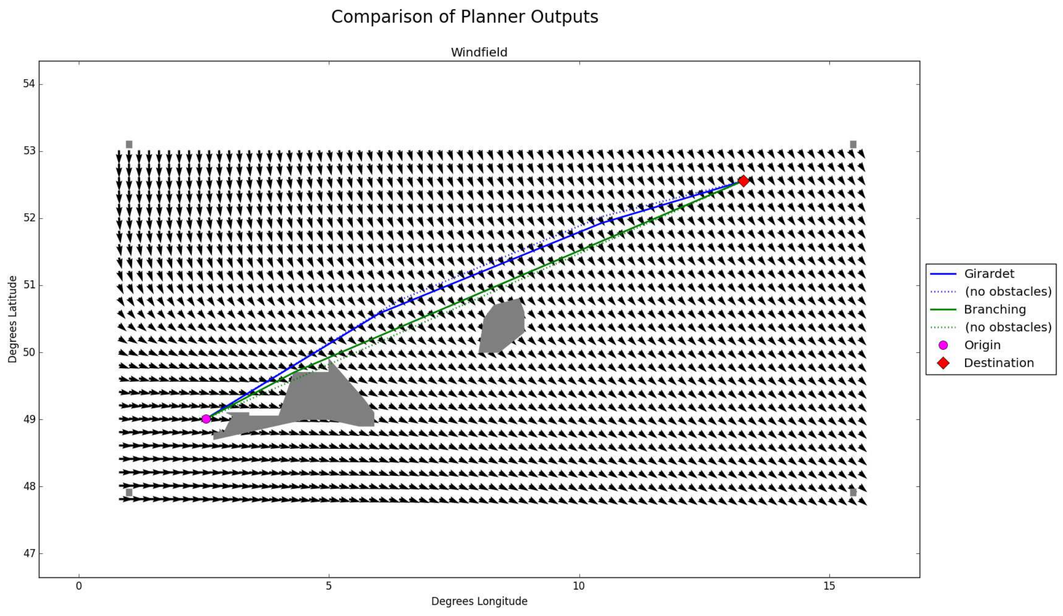


(e)

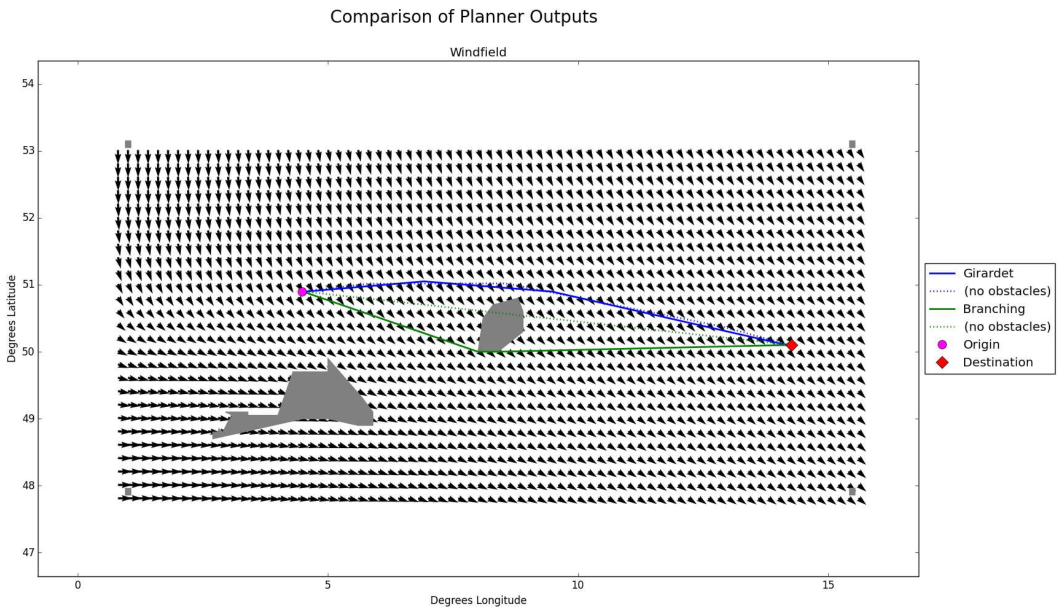


(f)

Figure D-4: (continued) Graphical Results for Medium Windfield Complexity, Low Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

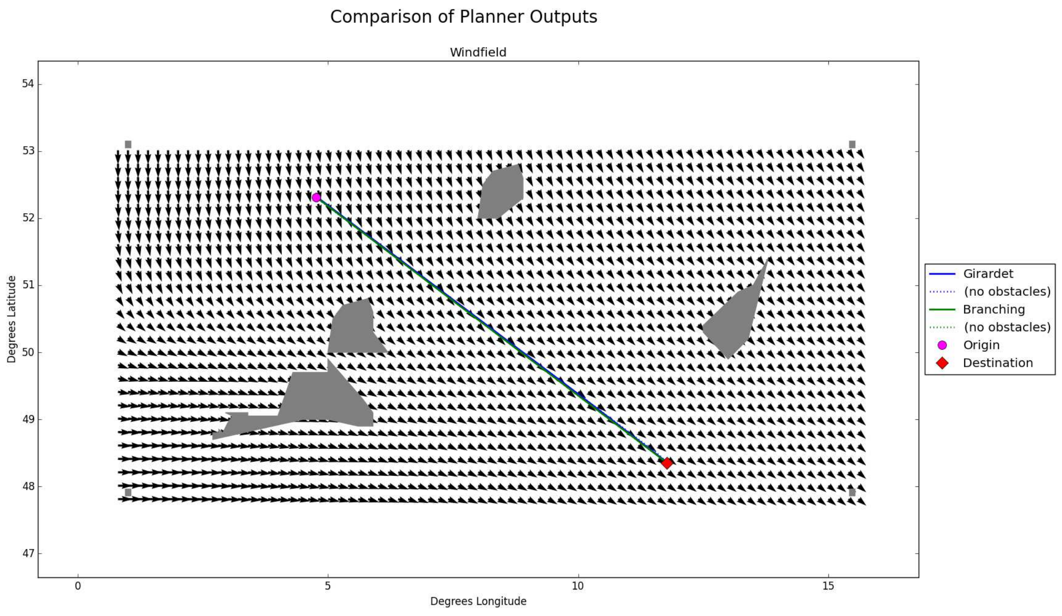


(g)

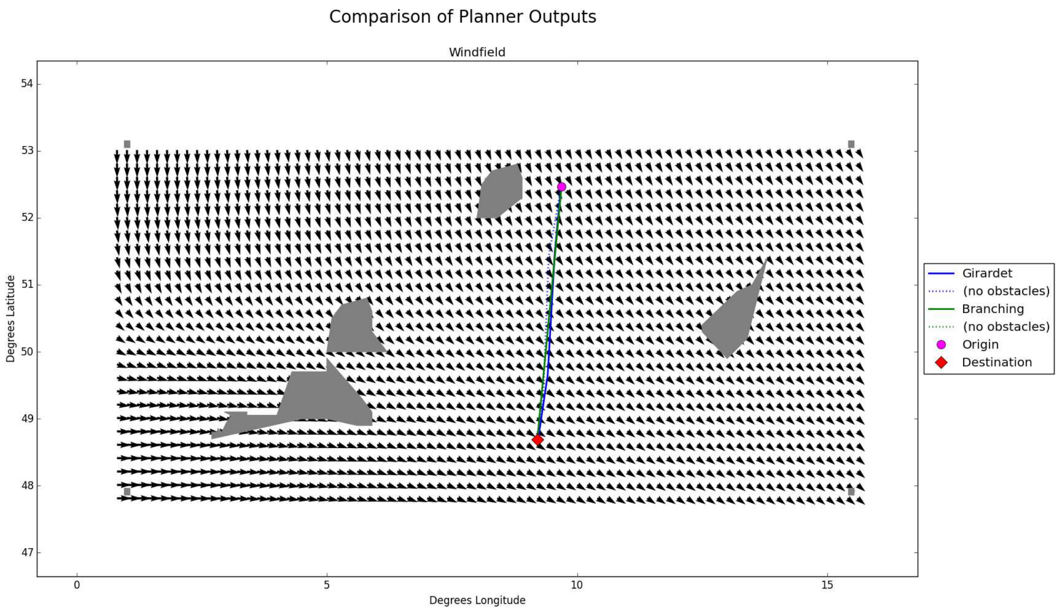


(h)

Figure D-4: (continued) Graphical Results for Medium Windfield Complexity, Low Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

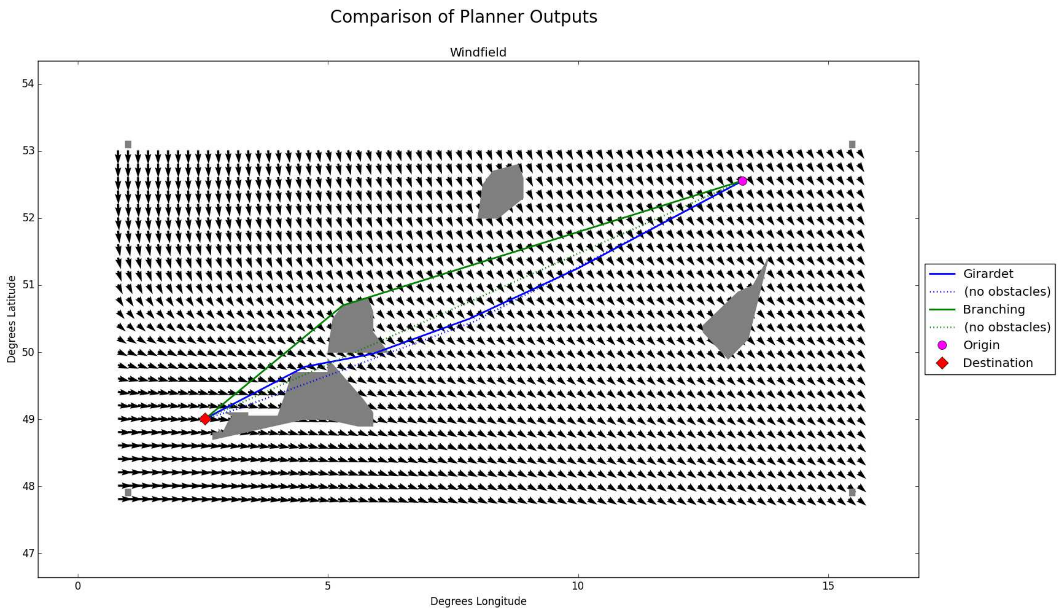


(a)

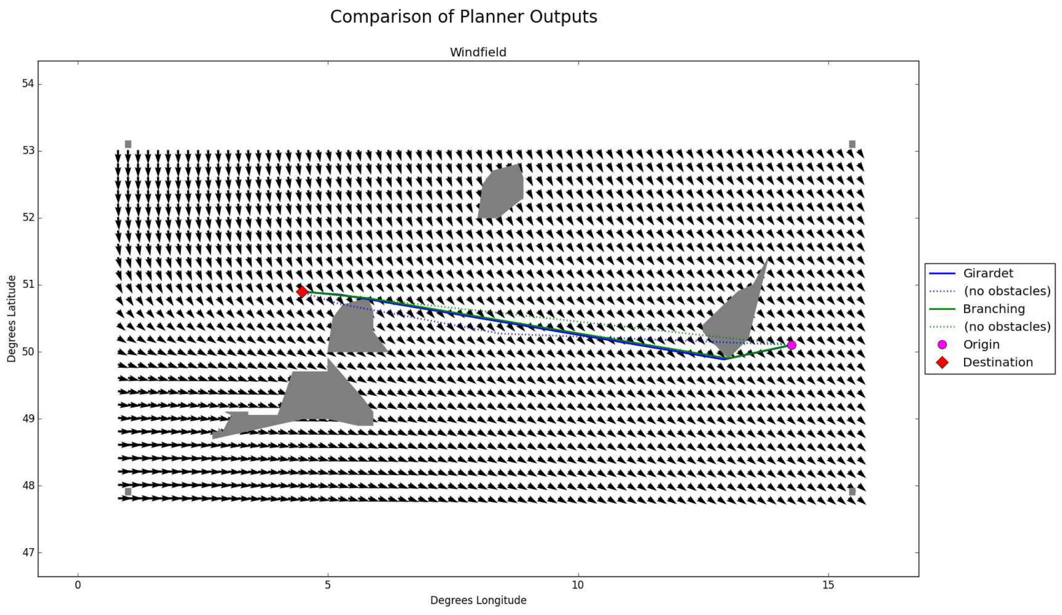


(b)

Figure D-5: Graphical Results for Medium Windfield Complexity, Medium Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

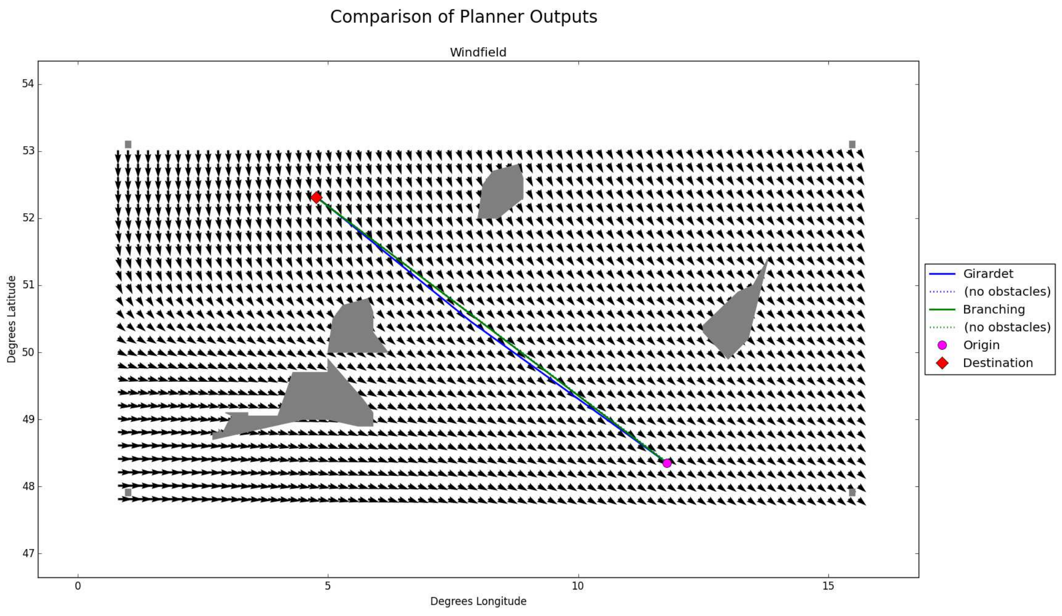


(c)

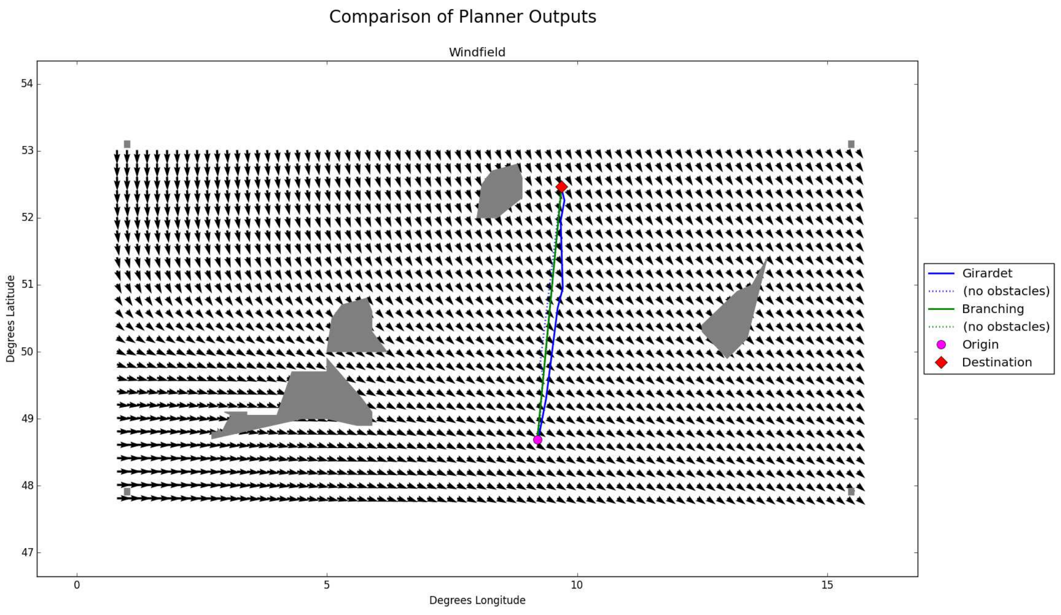


(d)

Figure D-5: (continued) Graphical Results for Medium Windfield Complexity, Medium Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

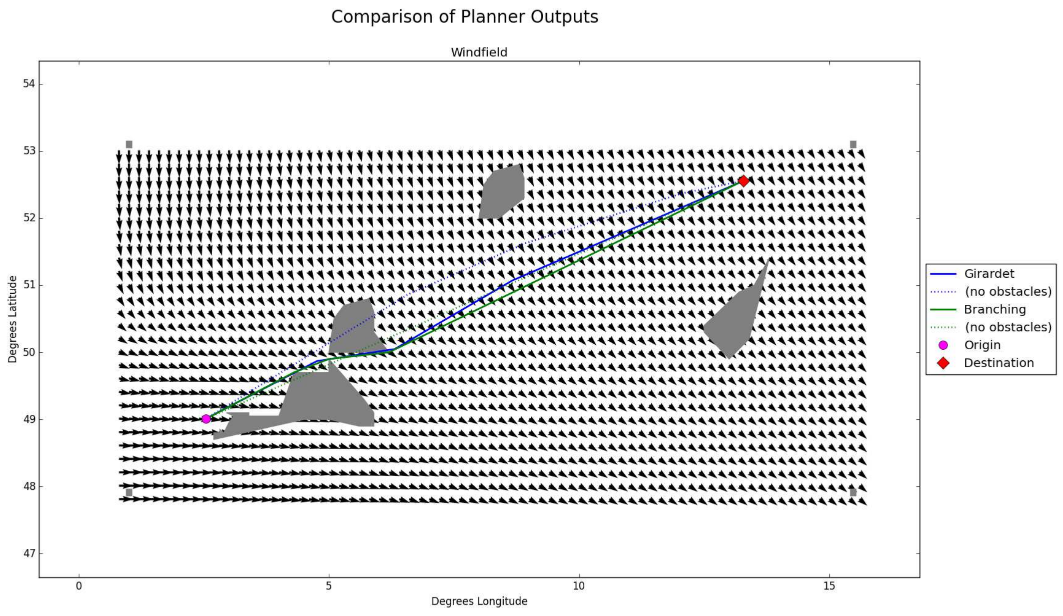


(e)

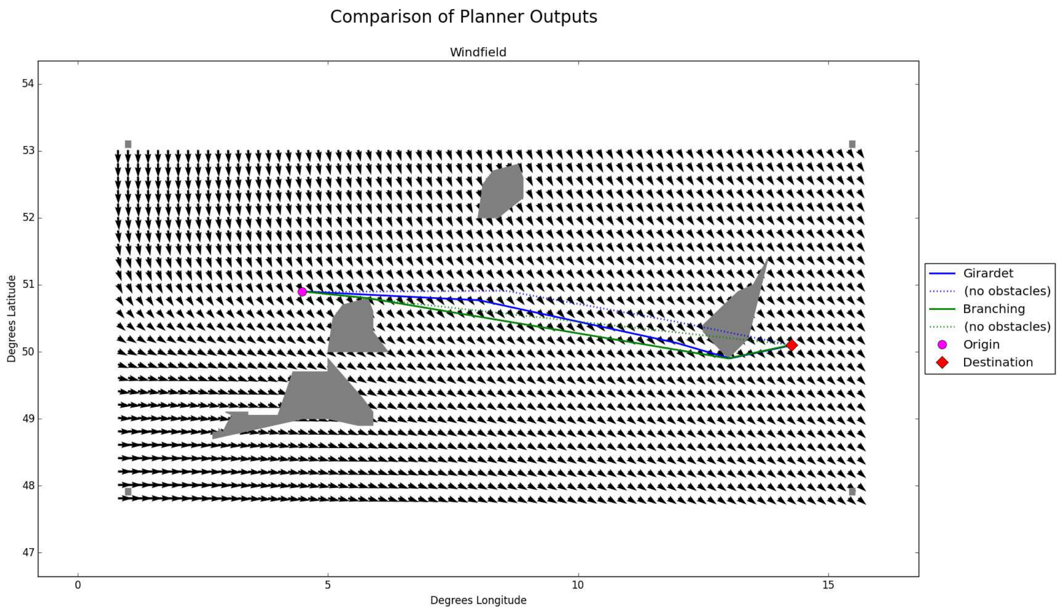


(f)

Figure D-5: (continued) Graphical Results for Medium Windfield Complexity, Medium Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

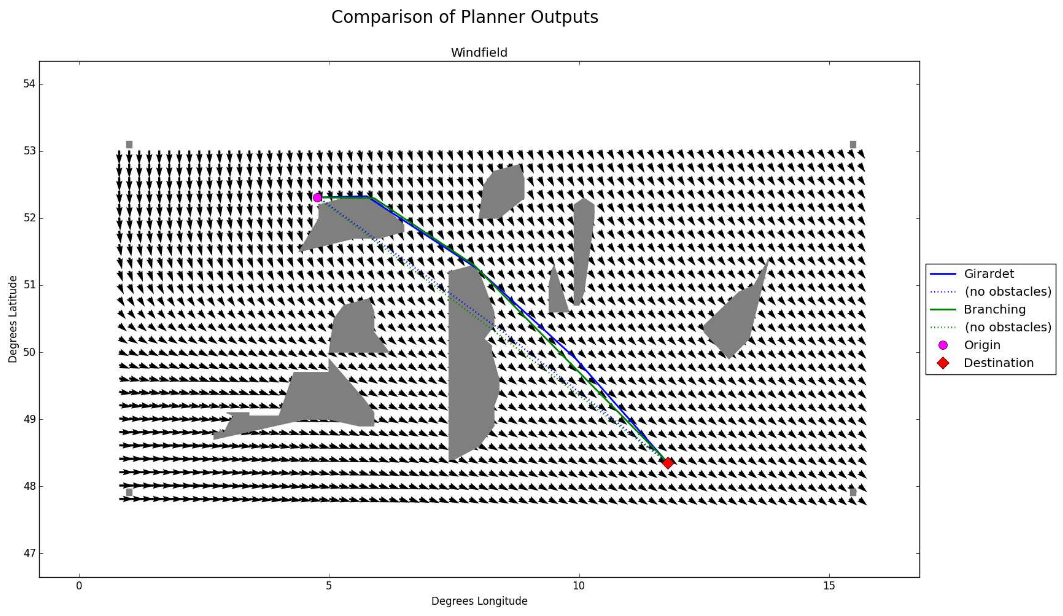


(g)

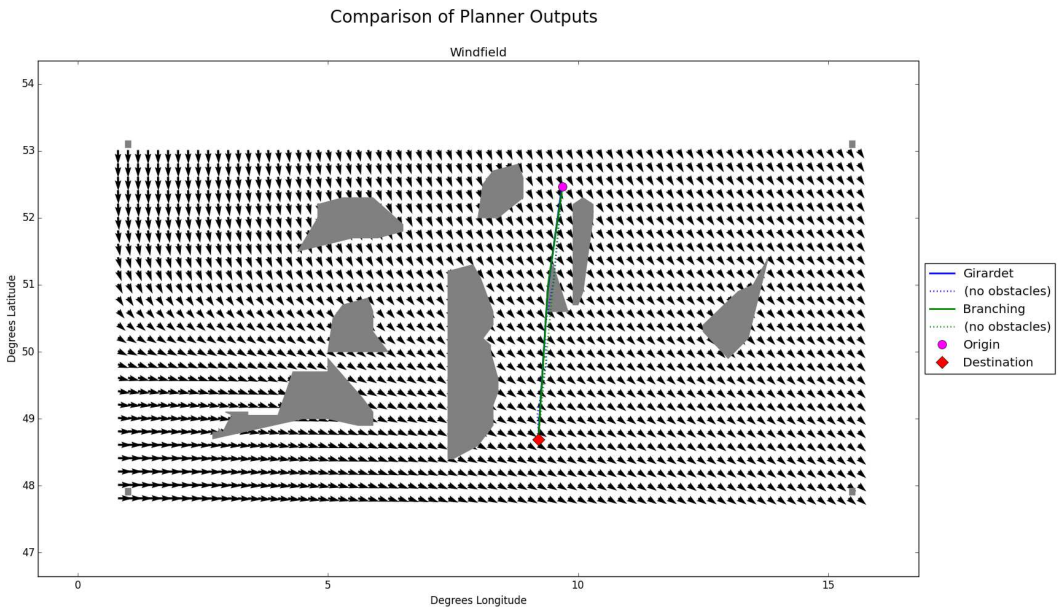


(h)

Figure D-5: (continued) Graphical Results for Medium Windfield Complexity, Medium Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

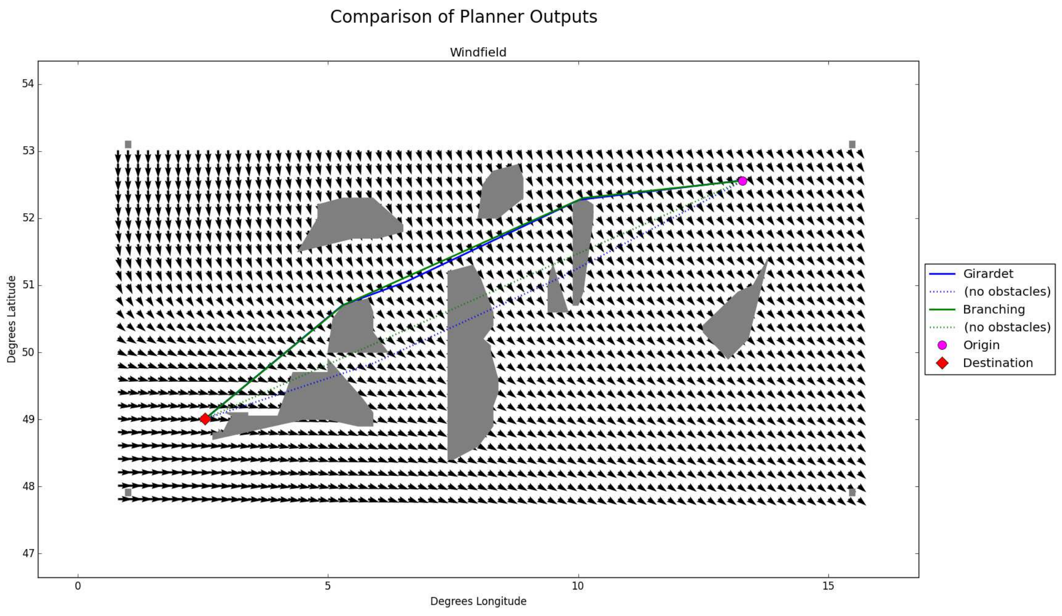


(a)

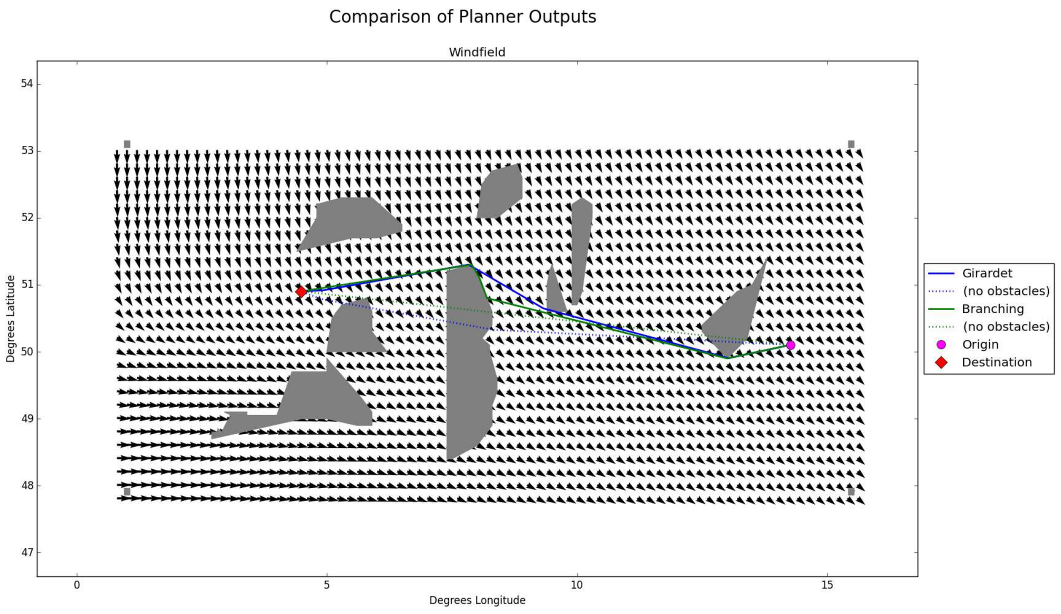


(b)

Figure D-6: Graphical Results for Medium Windfield Complexity, High Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

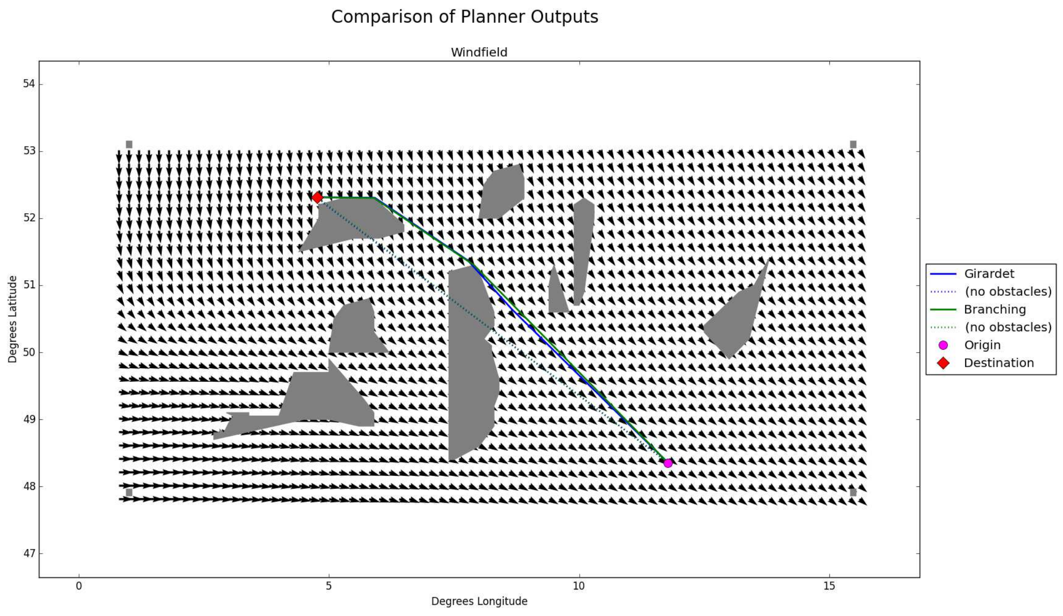


(c)

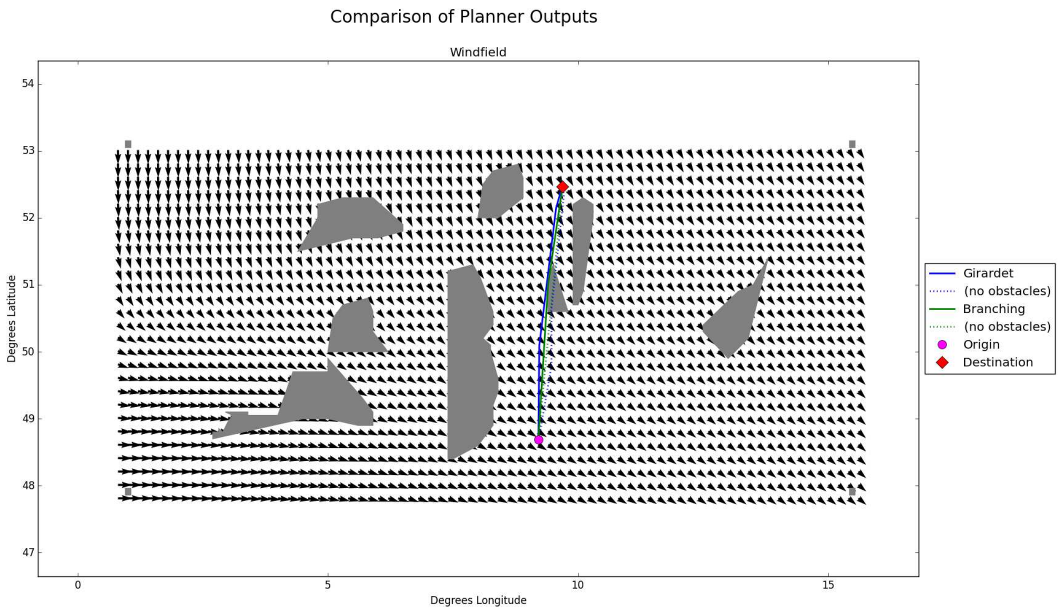


(d)

Figure D-6: (continued) Graphical Results for Medium Windfield Complexity, High Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

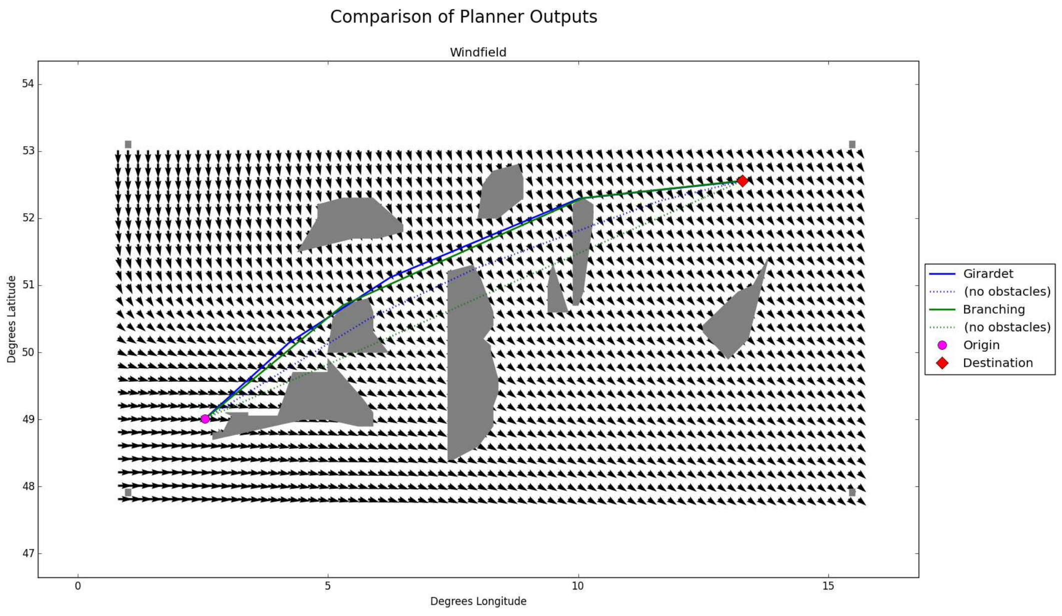


(e)

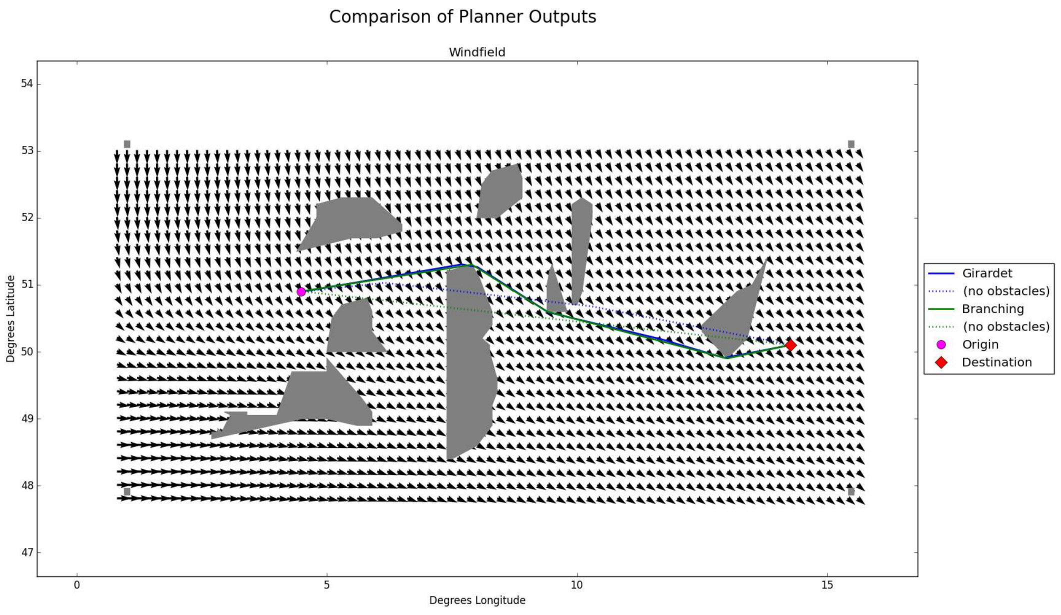


(f)

Figure D-6: (continued) Graphical Results for Medium Windfield Complexity, High Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

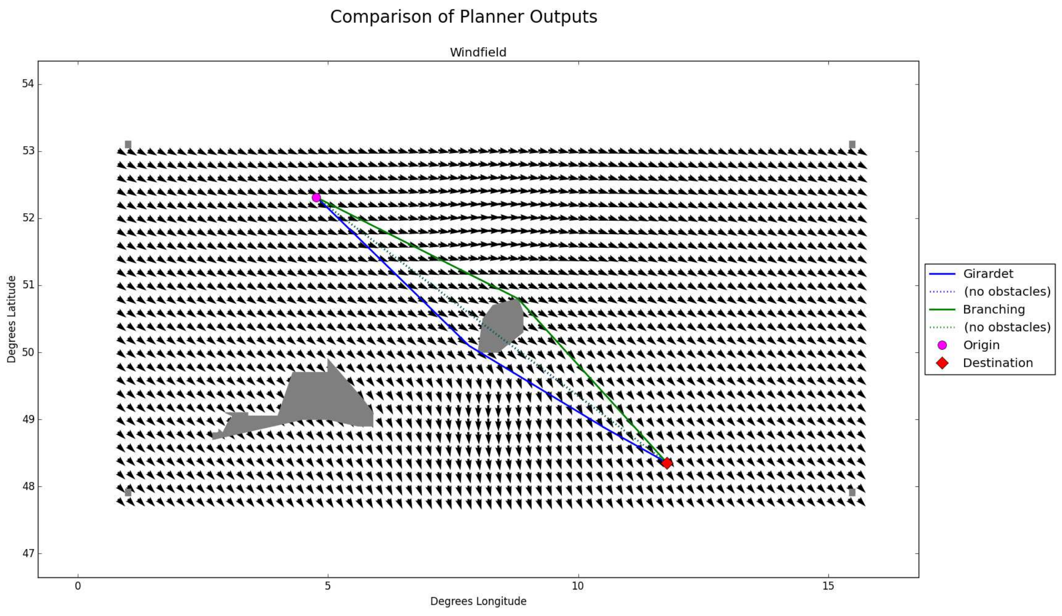


(g)

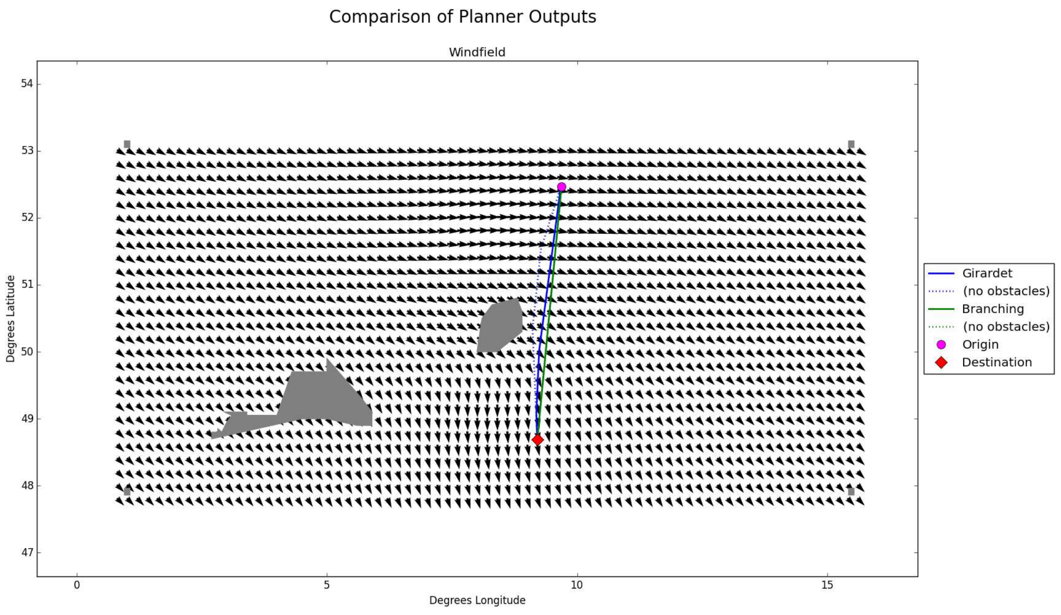


(h)

Figure D-6: (continued) Graphical Results for Medium Windfield Complexity, High Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

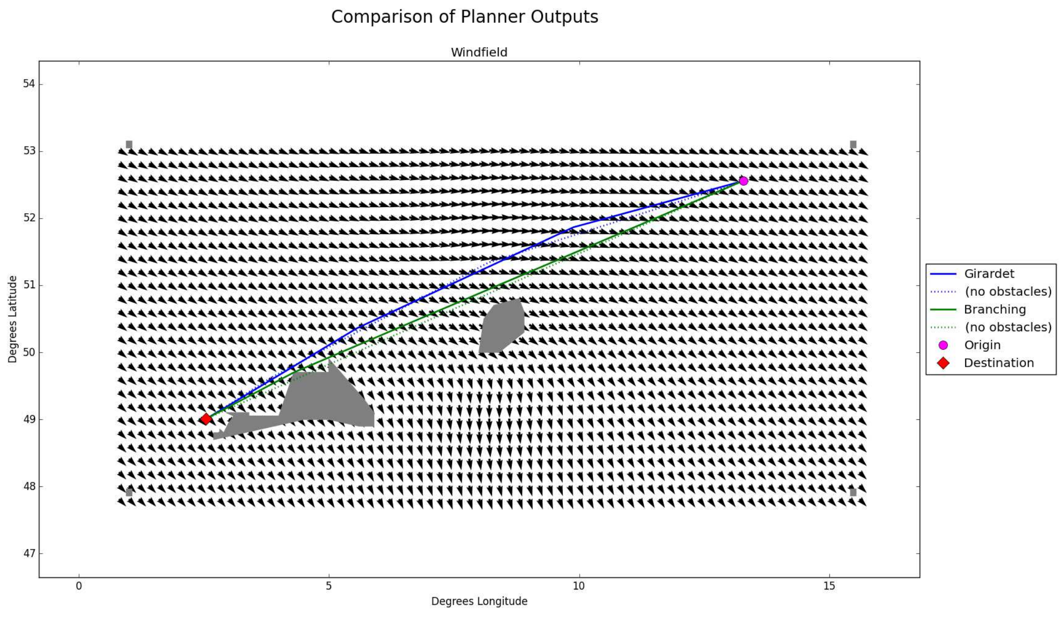


(a)

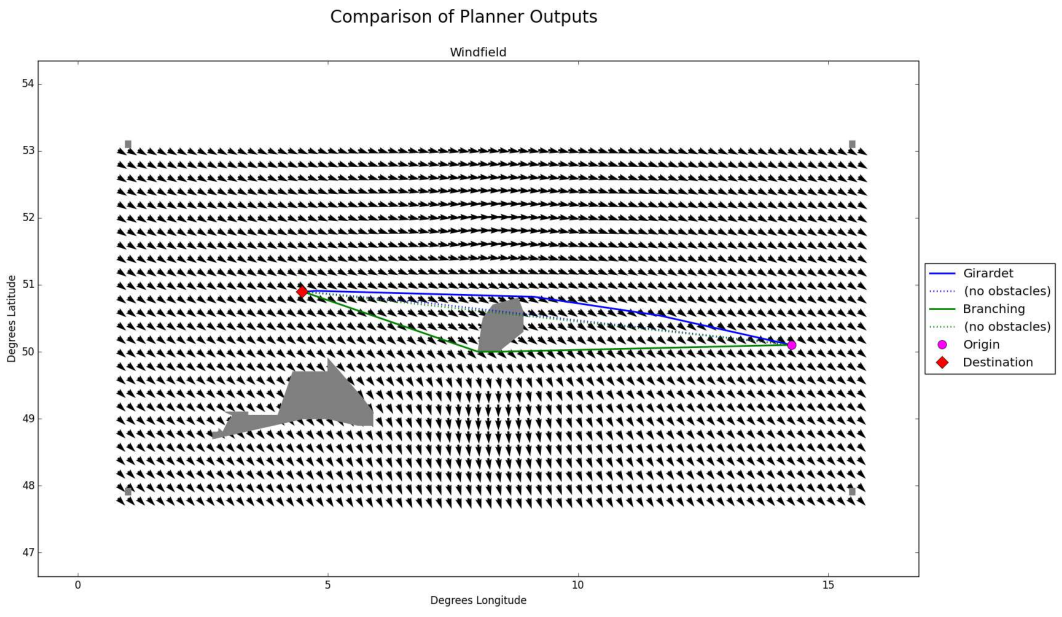


(b)

Figure D-7: Graphical Results for High Windfield Complexity, Low Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

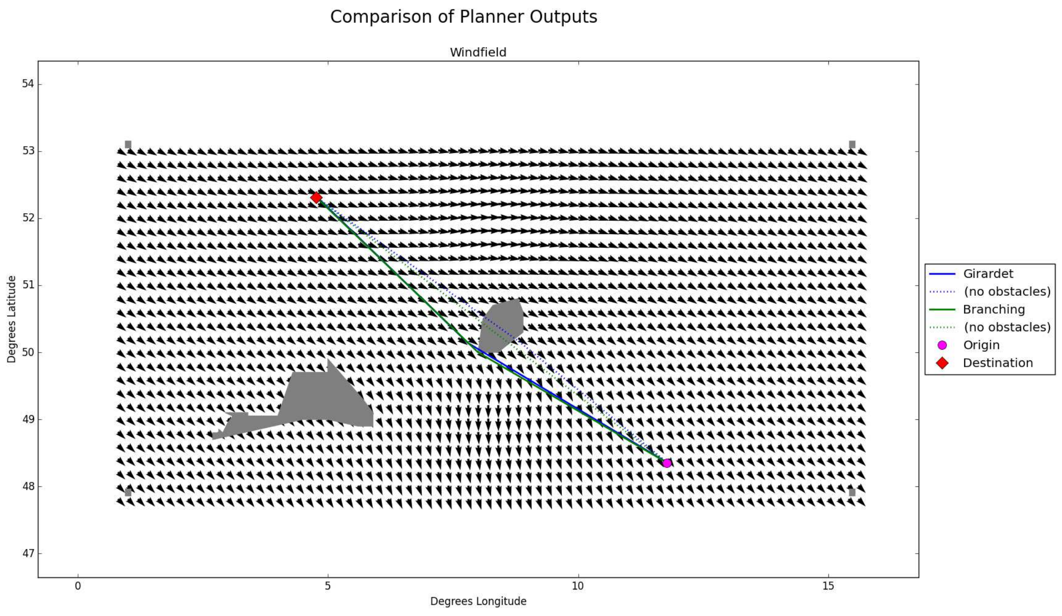


(c)

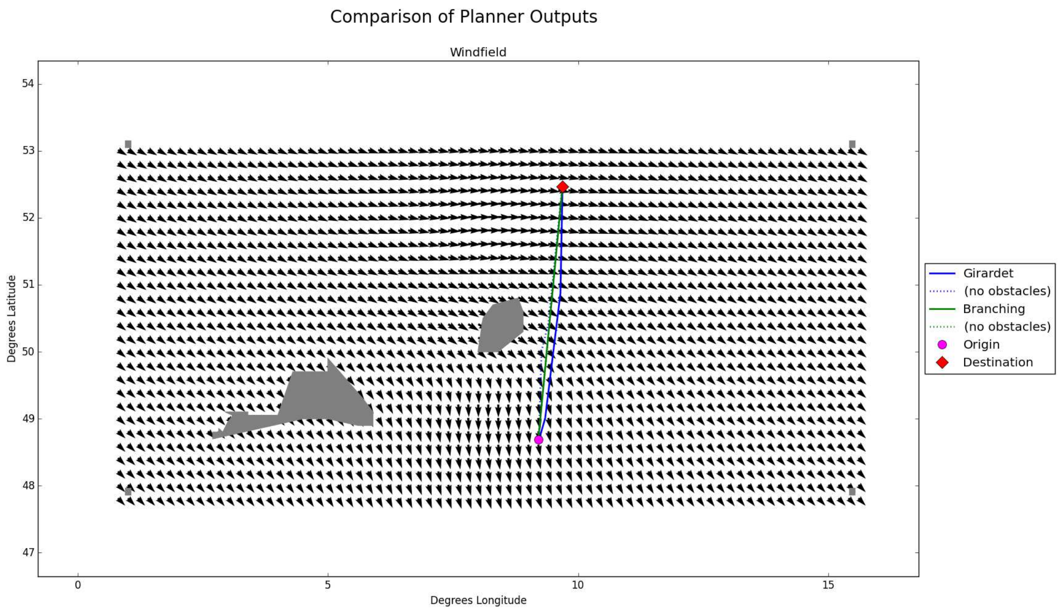


(d)

Figure D-7: (continued) Graphical Results for High Windfield Complexity, Low Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

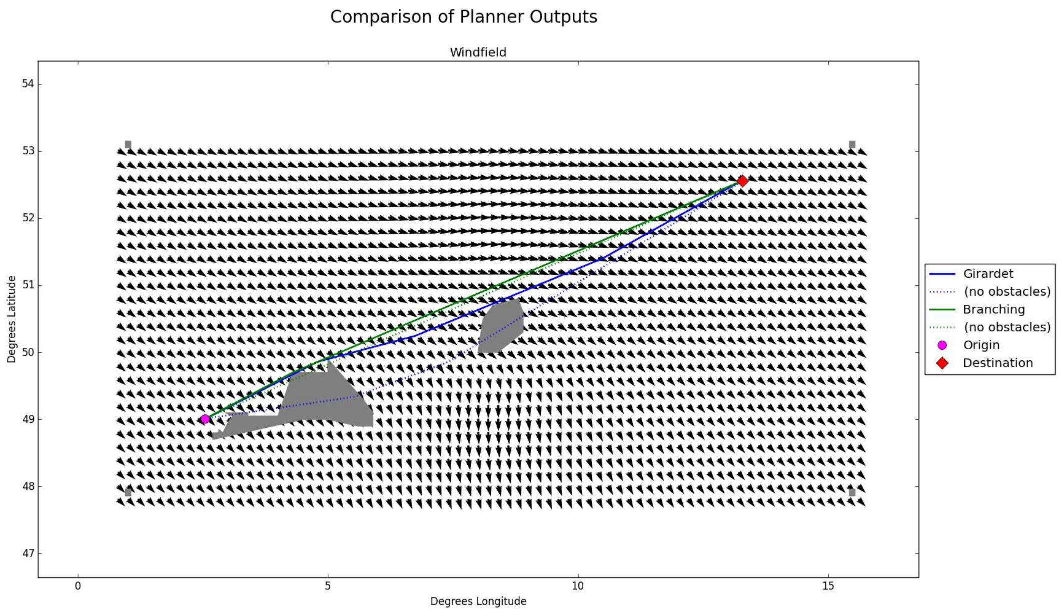


(e)

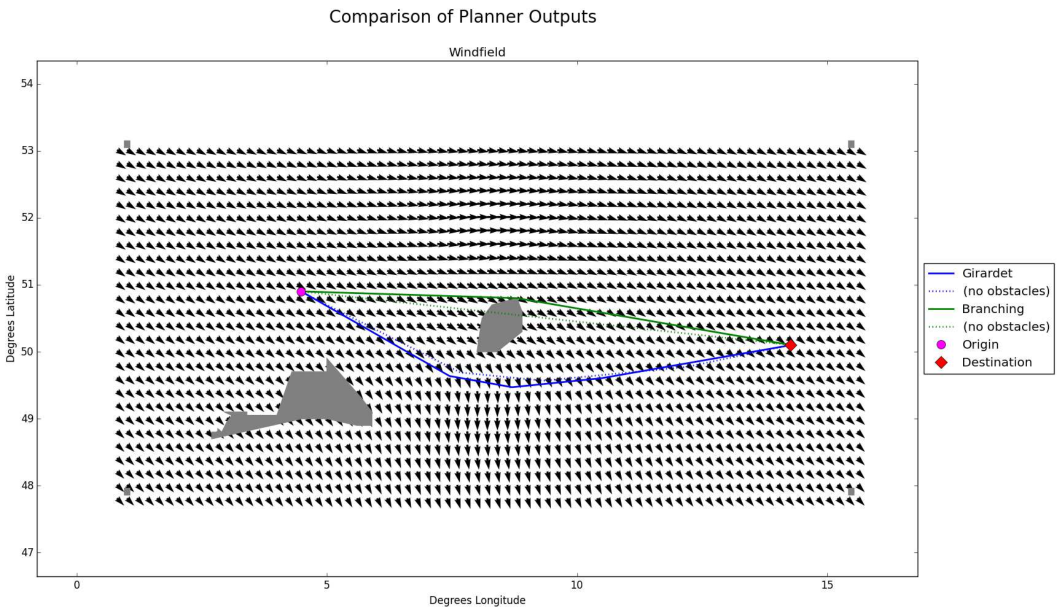


(f)

Figure D-7: (continued) Graphical Results for High Windfield Complexity, Low Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

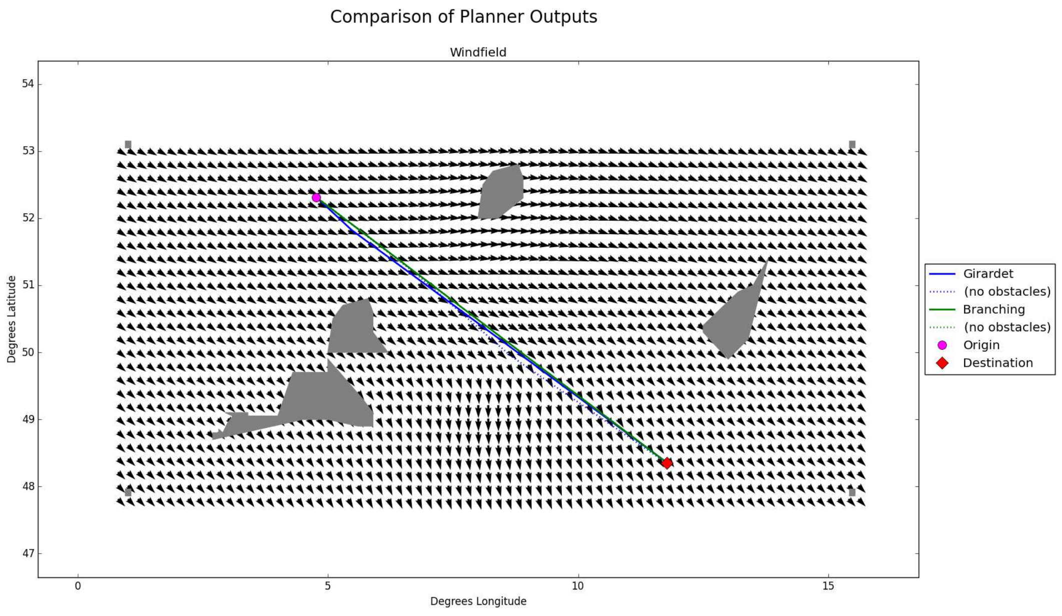


(g)

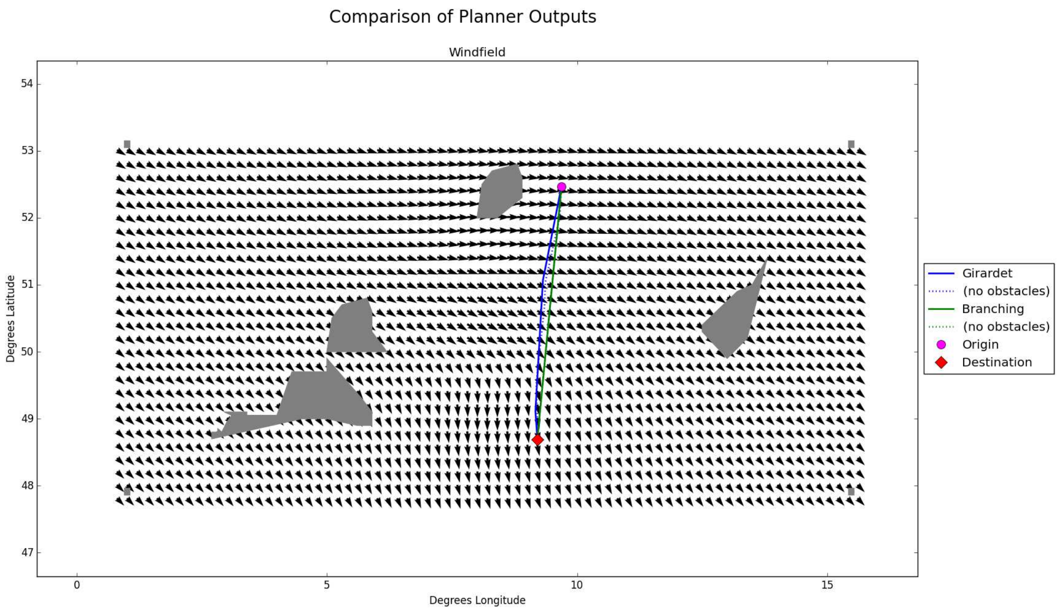


(h)

Figure D-7: (continued) Graphical Results for High Windfield Complexity, Low Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

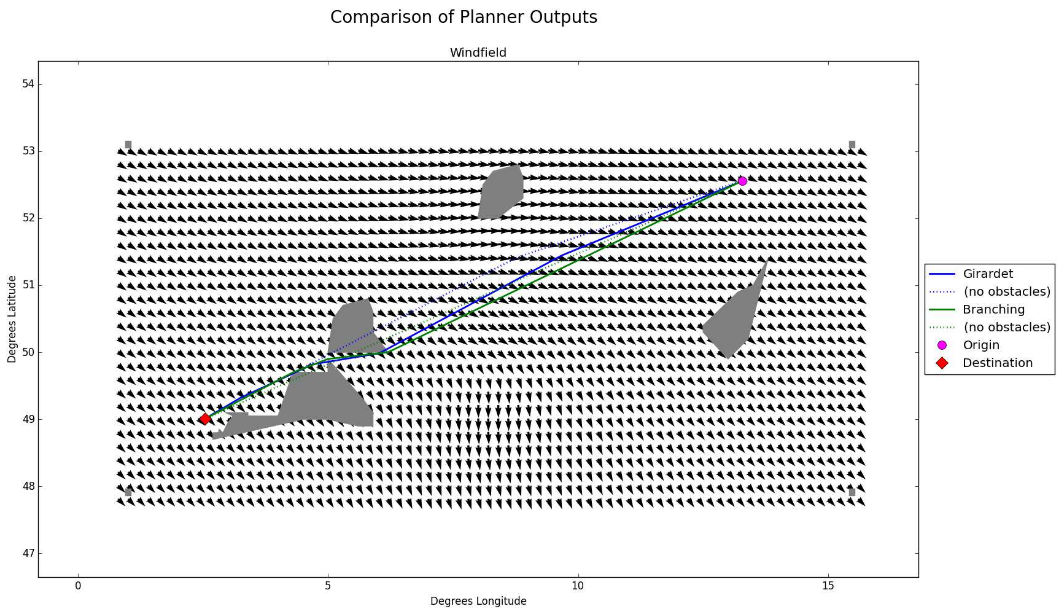


(a)

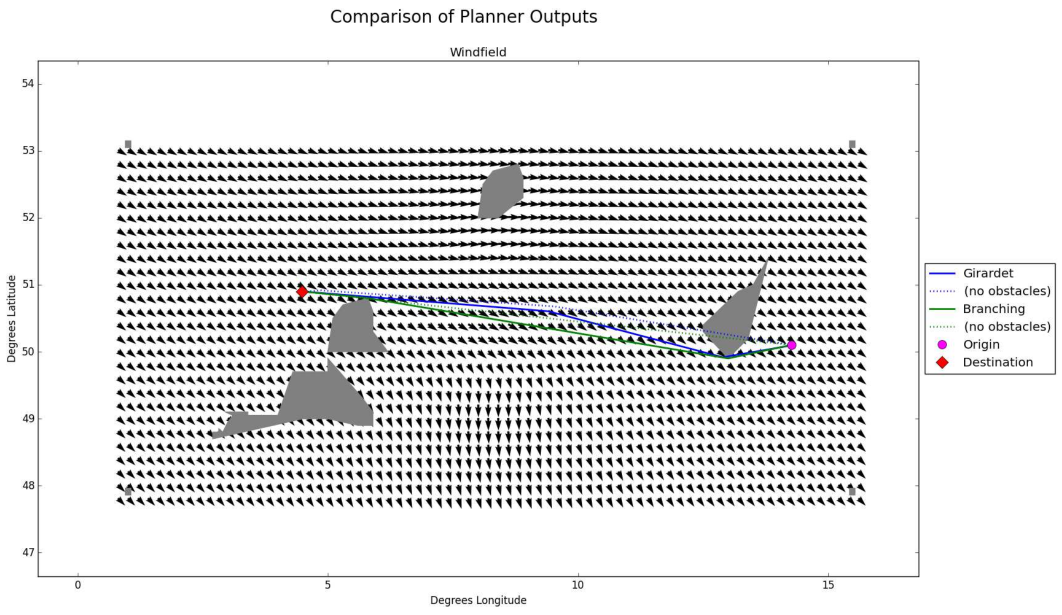


(b)

Figure D-8: Graphical Results for High Windfield Complexity, Medium Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

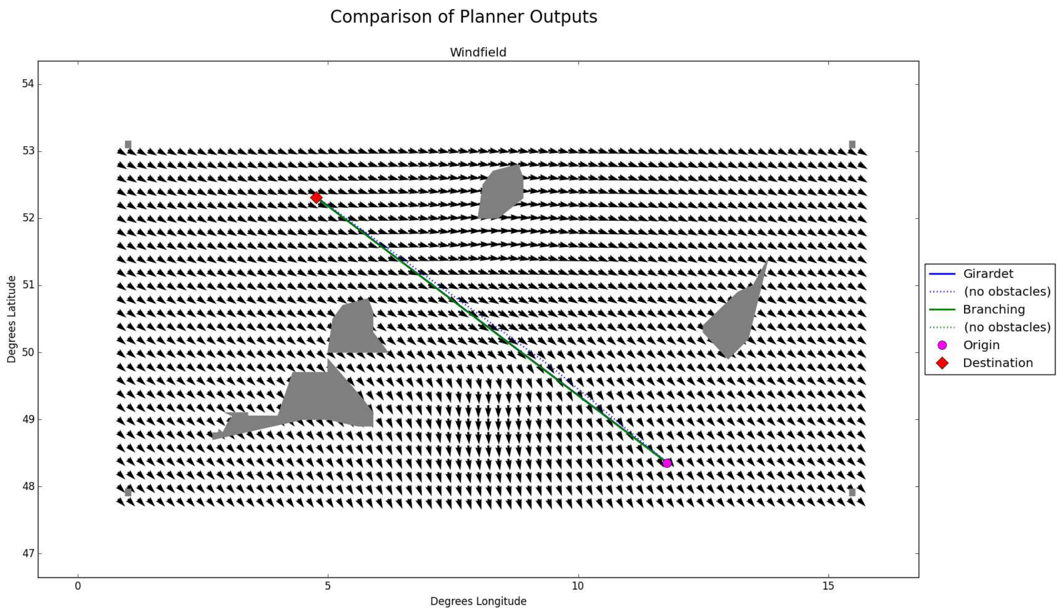


(c)

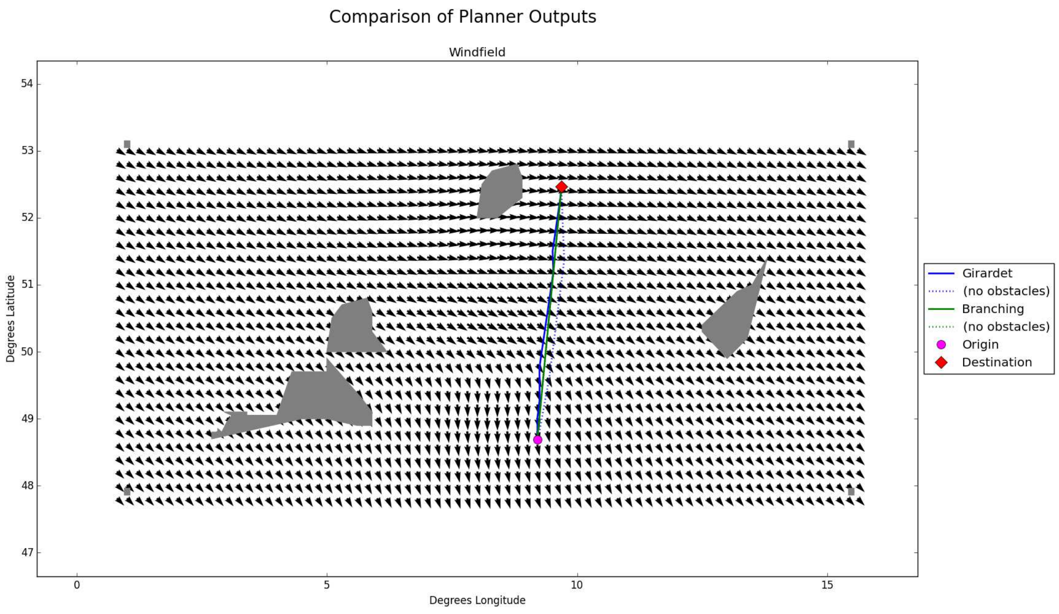


(d)

Figure D-8: (continued) Graphical Results for High Windfield Complexity, Medium Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

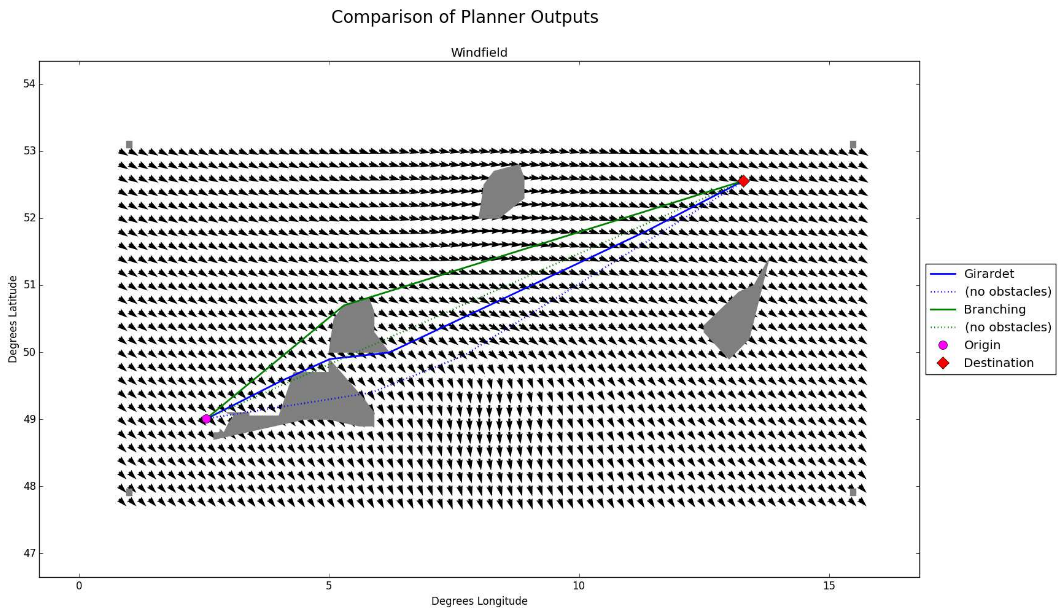


(e)

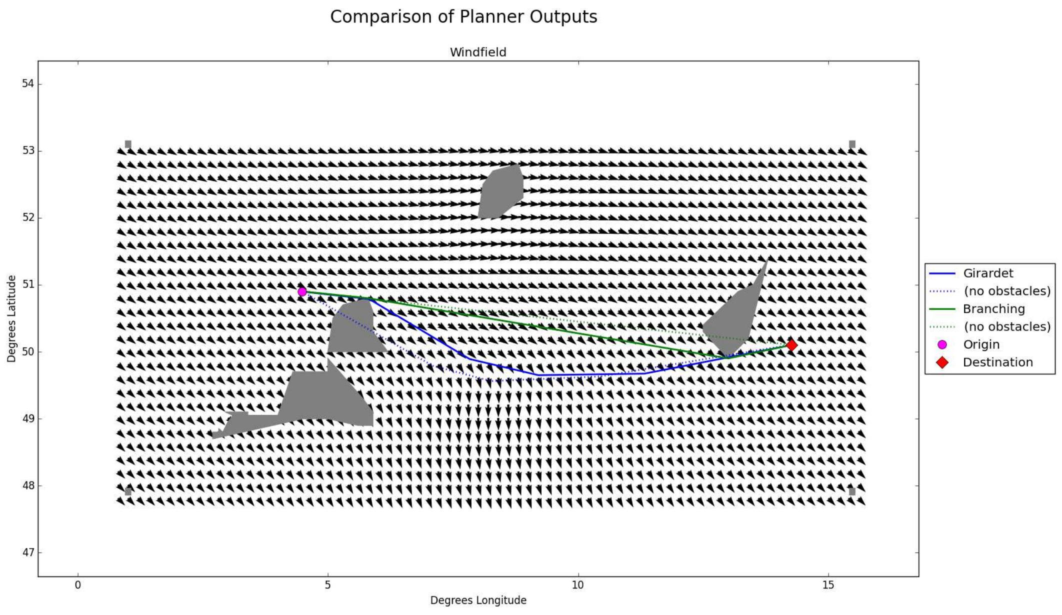


(f)

Figure D-8: (continued) Graphical Results for High Windfield Complexity, Medium Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

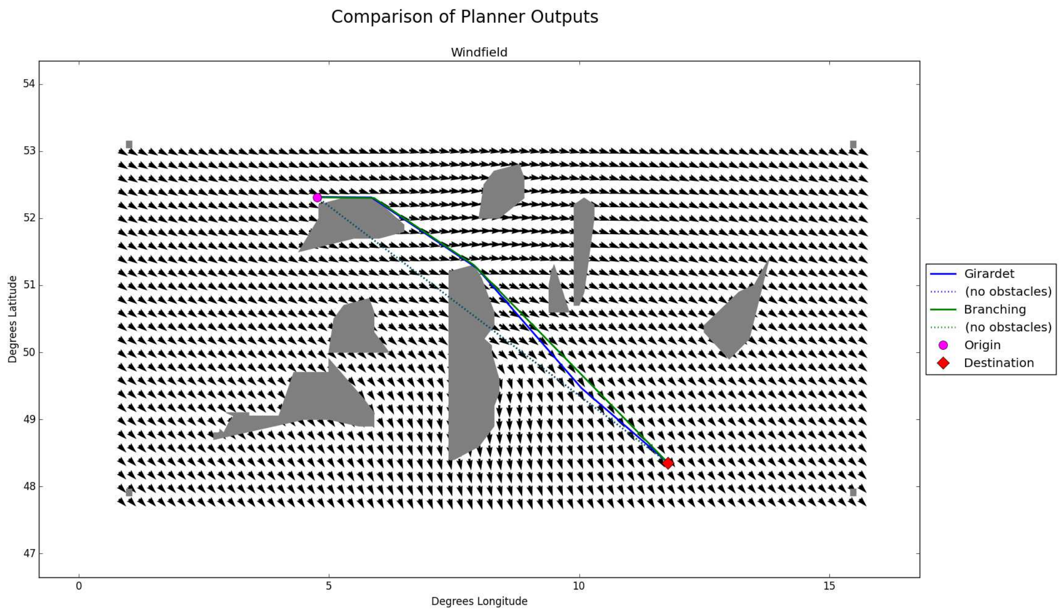


(g)

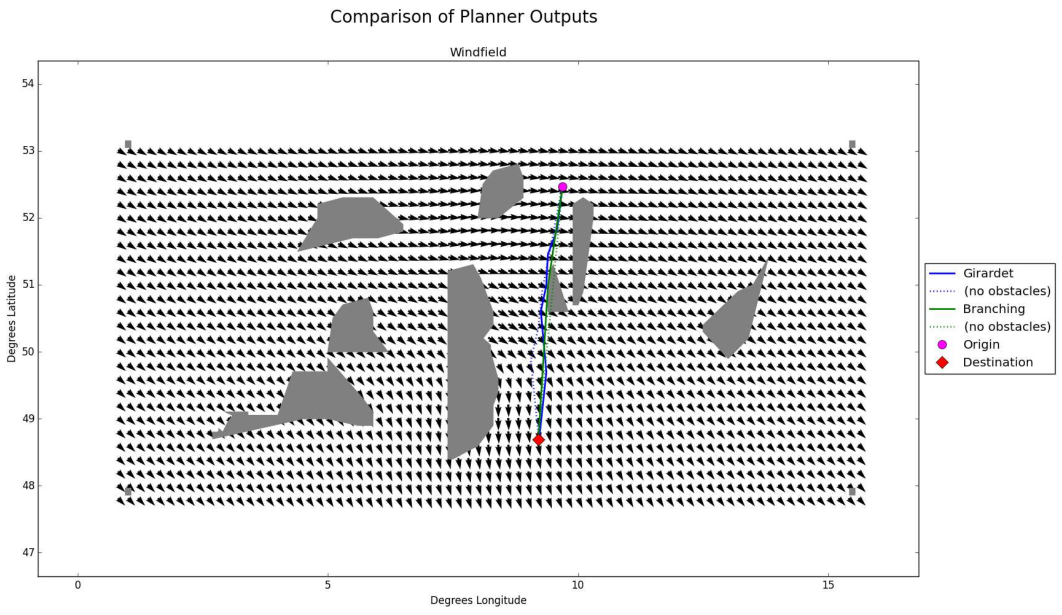


(h)

Figure D-8: (continued) Graphical Results for High Windfield Complexity, Medium Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

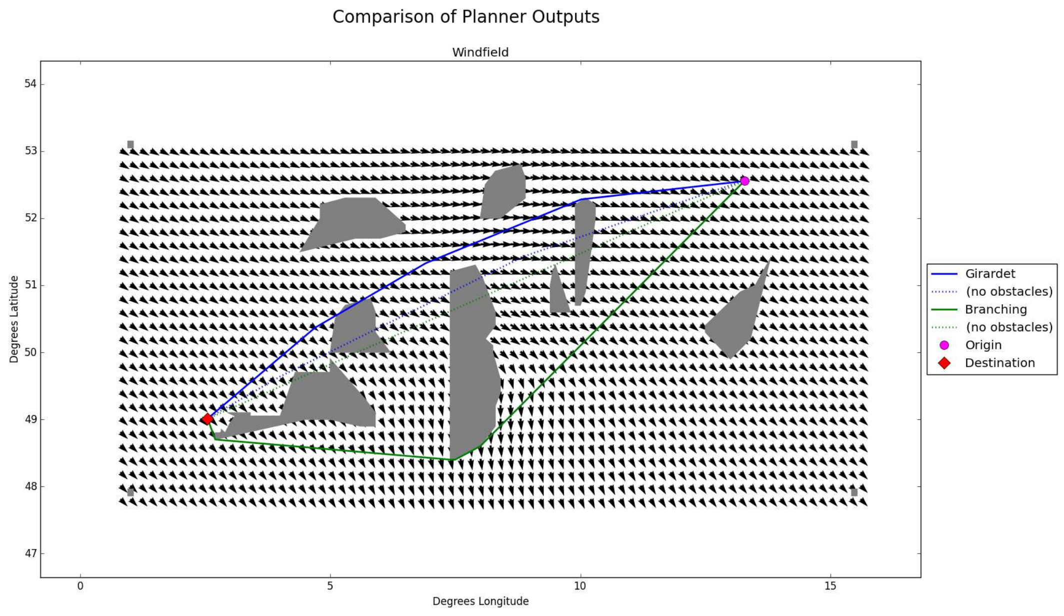


(a)

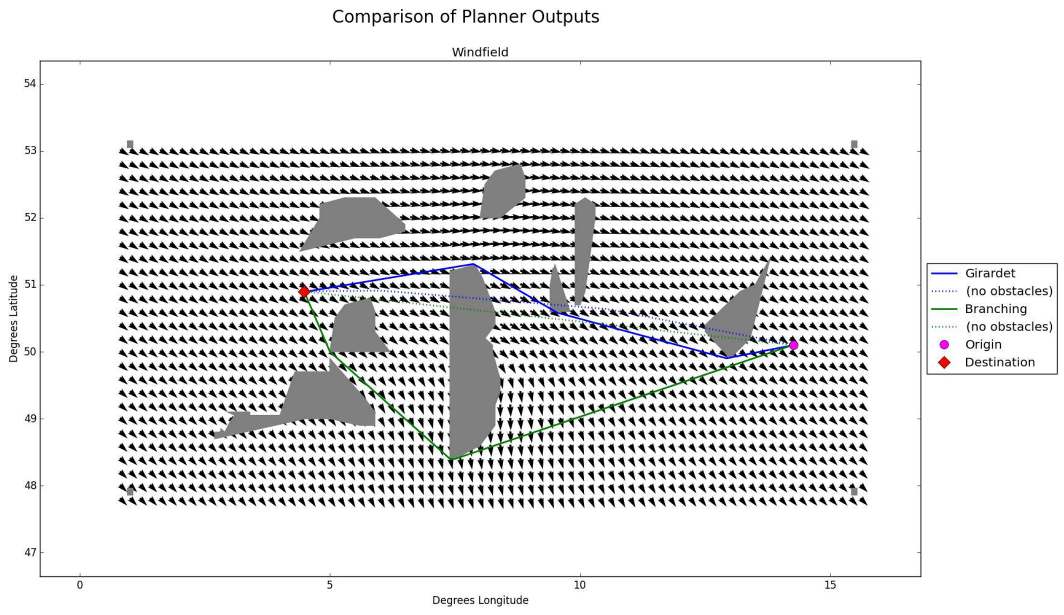


(b)

Figure D-9: Graphical Results for High Windfield Complexity, High Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

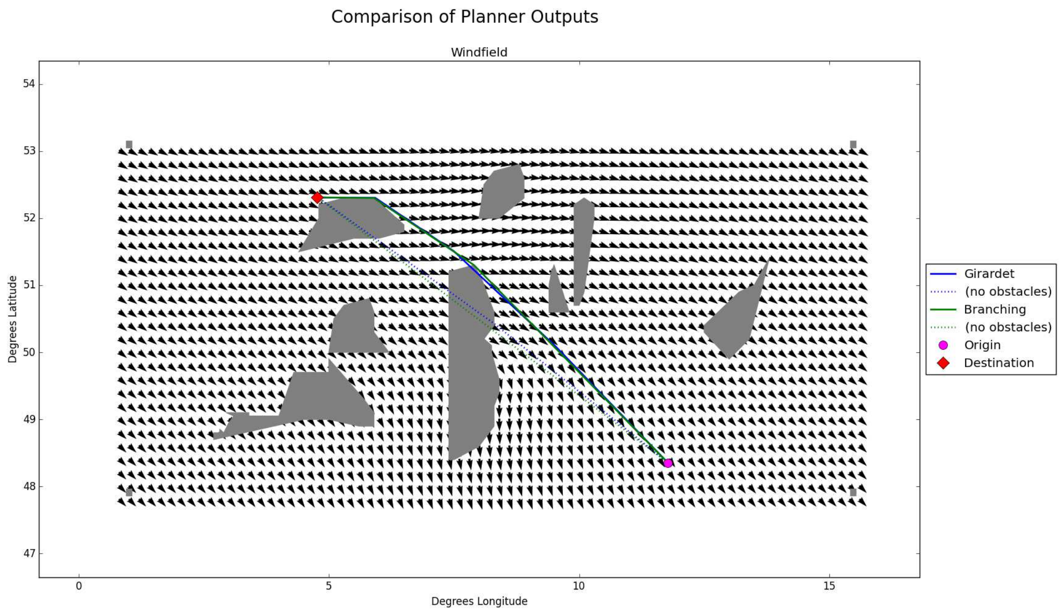


(c)

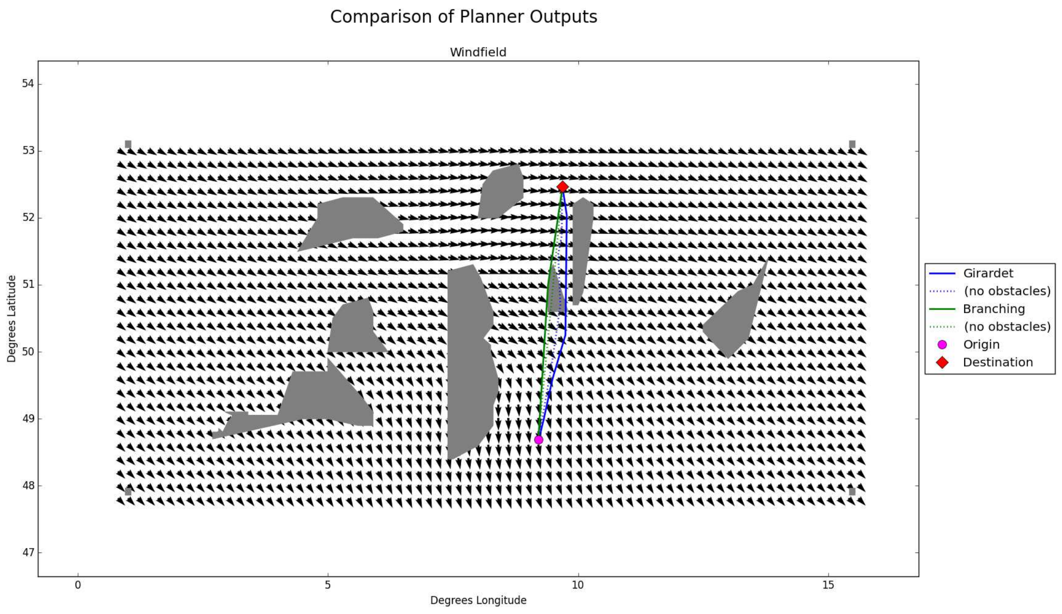


(d)

Figure D-9: (continued) Graphical Results for High Windfield Complexity, High Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

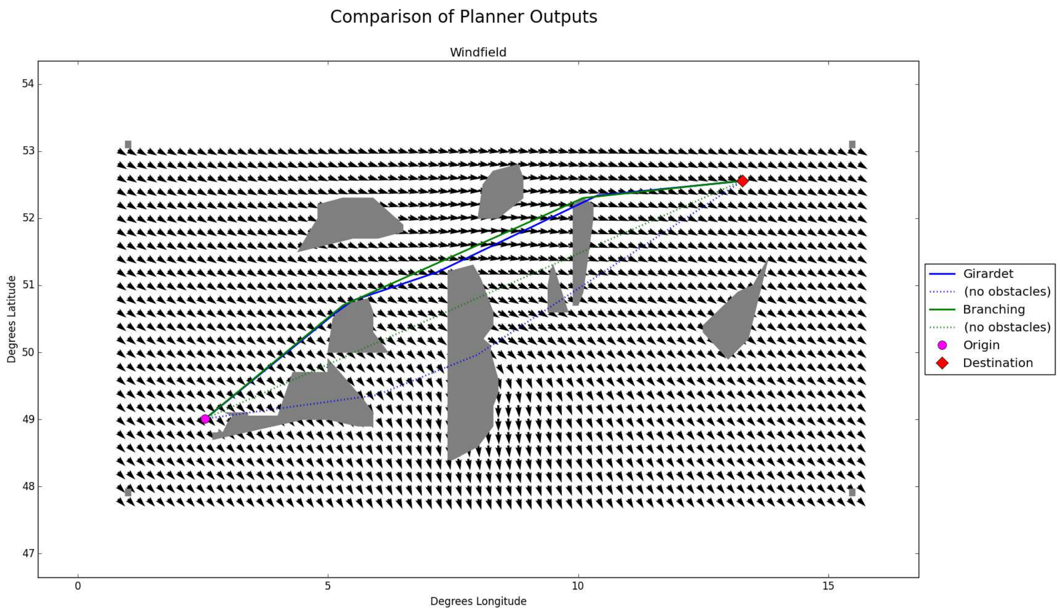


(e)

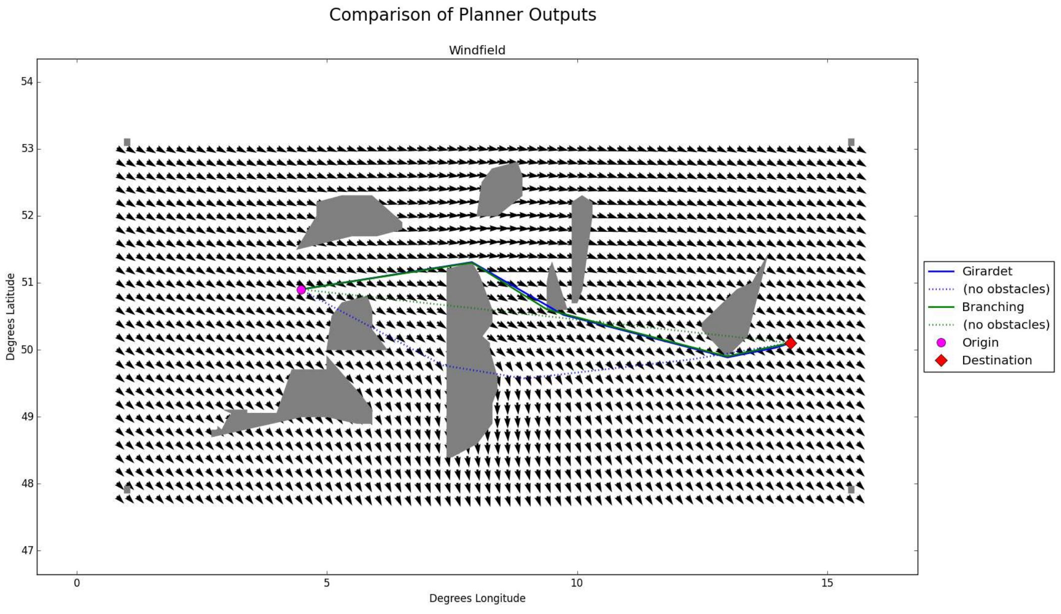


(f)

Figure D-9: (continued) Graphical Results for High Windfield Complexity, High Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)



(g)



(h)

Figure D-9: (continued) Graphical Results for High Windfield Complexity, High Obstacle Field Complexity Condition: City Pair Repetitions (**a**)–(**h**)

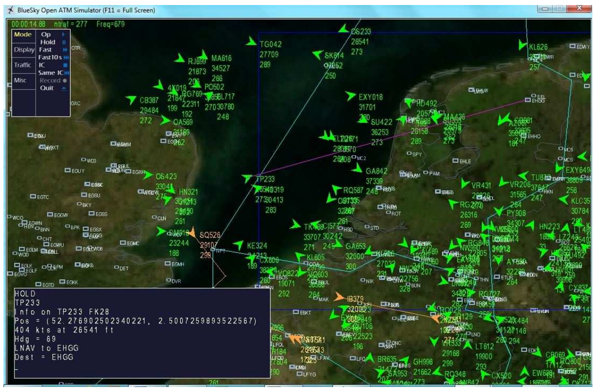
Appendix E

Research Continuation Recommendations

The recommendations for continued research that have been given in the technical paper of Part I are elaborated upon in this appendix.

One of the primary observations from this work is a potential to combine the strengths of the two planners. This combination can be achieved by finding the wind-optimal route without regard for obstacles using the Girardet planner and then making route adjustments to avoid obstacles using the Branching planner. Investigating the benefits and limitations of such a combined planner would be a useful research aim. Perhaps the simulation environment can be extended to BlueSky, which is the open-access ATM research tool being developed at Delft University of Technology (DUT). Since BlueSky is a Python-based software, a transition from the current implementations of the planners to one within BlueSky might be readily achievable. Aside from the benefit of a realistic environment visualization, as exemplified by the BlueSky screenshot provided in Figure E-1, the future positions of traffic could be considered as obstacles to avoid as well. The exclusion of traffic from the obstacle field considered by the Branching planner was one of the simplifications made to scope the current research project (as was shown in Appendix C). Although traffic is not a static obstacle, the location of traffic can be projected within a TBO environment, and thus is a realistic candidate for inclusion in the obstacle field considered by the 4DT planner.

A further limitation of the current research is the constant-altitude assumption. It is expected that the logic of the Branching planner will generalize nicely for the addition of the altitude dimension. Specifically, the planner could make trial solutions for “above” and “below”, just as it currently does for “clockwise” and “counterclockwise” rotations about obstacles. Combinations of these maneuvers will be achieved by the result of the backward cleanup step of the algorithm presented in the technical paper of Part I. It is therefore suggested that the backward cleanup step be improved: in the current implementation, the backward cleanup is limited to simply removing unnecessary waypoints from the path that is under consideration (and there is even one example of the backward cleanup step failing to remove an unnecessary waypoint in Figure D-6d). A waypoint is deemed unnecessary if the removal of the waypoint



```
HOLD
TP233
Info on TP233 FK28
Pos = (52.276902502340221, 2.5007259893522567)
404 kts at 26541 ft
Hdg = 69
LNAV to EHGG
Dest = EHGG
_
```

Figure E-1: BlueSky Screenshot (Hoekstra, 2015)

creates a route which does not intersect with any obstacle. This backward cleanup method is thus a conservative method, since it is possible to further optimize the route between two points with a curvature of the route. This mention of curvature raises the issue of including constraints on dynamics, such as those limitations known to the flight management system, in the Branching planner logic. The output of the planner could therefore be made dynamically feasible, but it would perhaps be more efficient to incorporate a dynamic feasibility check at each branching decision such that the outcome will necessarily be dynamically feasible.

Bibliography

Ashford, R. (2010). *NextGen Trajectory-Based Operations Status Update.* (NASA Report)

Ballin, M. G., Sharma, V., Vivona, R. A., Johnson, E. J., & Ramiscal, E. (2002). A Flight Deck Decision Support Tool for Autonomous Airborne Operations. In *AIAA Guidance, Navigation, and Control Conference.* Monterey.

Baras, J., & Theodorakopoulos, G. (2010). *Path Problems in Networks.* Breinigsville, Pennsylvania: Morgan & Claypool Publishers.

Bussink, F. J., Verhoeven, R., Marsman, A., Prats, X., Bendris, B., Montolio, J., et al. (2016). Optimization of the vertical trajectory through Time and Energy management: A Human-in-the-Loop Study. In *AIAA Guidance, Navigation, and Control Conference.* San Diego: AIAA SciTech.

Cate, K. T. (2013, January). Challenges in Achieving Trajectory-Based Operations. In *51st AIAA Aerospace Sciences Meeting* (pp. 1–6). Dallas/Ft. Worth.

Chaimatanan, S., Delahaye, D., & Mongeau, M. (2012). A methodology for Strategic Planning of Aircraft Trajectories using Simulated Annealing. In *1st International Conference on Interdisciplinary Science for Air Traffic Management.* Daytona Beach.

Da Silva, S. (2012). Trajectory-Based Operations(TBO). In *ICAO SIP 2012 - ASBU workshops.* Bangkok.

Dalmau, R., & Prats, X. (2014). How much fuel can be saved in a perfect flight trajectory? Continuous cruise climbs vs. conventional operations. In *6th International Congress on Research in Air Transportation (ICRAT).* Istanbul.

Eele, A., & Richards, A. (2009). Path-Planning with Avoidance Using Nonlinear Branch-and-Bound Optimization. *Journal of Guidance, Control, and Dynamics*, *32*(2).

Enea, G., & Porretta, M. (2012). A Comparison of 4D-Trajectory Operations Envisioned for NextGen and SESAR, Some Preliminary Findings. In *28th International Congress of the Aeronautical Sciences* (pp. 1–14). Brisbane.

Erzberger, H. (2006). Automated Conflict Resolution for Air Traffic Control. In *25th International Congress of the Aeronautical Sciences.* Hamburg.

Girardet, B. (2014). *Trafic Aérien: Détermination optimale et globale des trajectoires d'avions en présence de vent.* (Ph.D. Dissertation)

Girardet, B., Lapasset, L., Delahaye, D., Rabut, C., & Brenier, Y. (2013). Generating

Optimal Aircraft Trajectories with respect to Weather Conditions. In *2nd International Conference on Interdisciplinary Science for Innovative Air Traffic Management.* Toulouse.

Hillier, F. S., & Lieberman, G. L. (2011). *Introduction to Operations Research* (10th ed.). McGraw-Hill International Edition.

Hoekstra, J. (2015). *BlueSky Homepage: BlueSky Open Air Traffic Simulator.* Available from `http://homepage.tudelft.nl/7p97s/BlueSky/` (Accessed 15 March 2016)

Joint Planning and Development Office. (2007, June). *Concept of Operations for the Next Generation Air Transportation System Version 2.0.*

Joint Planning and Development Office. (2011). *Concept of Operations for the Next Generation Air Transportation System Version 3.2.*

Karlof, J. K. (Ed.). (2006). *Integer Programming: Theory and Practice.* Boca Raton: Taylor & Francis Group.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–680.

Koczo, S., & Wing, D. (2013). An Operational Safety and Certification Assessment of a TASAR EFB Application. In *32nd Digital Avionics Systems Conference.* East Syracuse: IEEE.

Reynolds, T. G., McPartland, M., Teller, T., & Troxel, S. (2015). Exploring Wind Information Requirements for Four Dimensional Trajectory-Based Operations. In *11th USA/Europe Air Traffic Management Research and Development Seminar.* Lisbon.

Rios, J., & Lohn, J. (2009, August). A Comparison of Optimization Approaches for Nationwide Traffic Flow Management. In *AIAA Guidance, Navigation, and Control Conference.* Chicago.

Ruiz, S., & Soler, M. (2015). Conflict pattern analysis under the consideration of optimal trajectories in the European ATM. In *11th USA/Europe Air Traffic Management Research and Development Seminar* (pp. 1–10). Lisbon.

Sathyaraj, B. M., Jain, L. C., Finn, A., & Drake, S. (2008). Multiple UAVs path planning algorithms: a comparative study. *Fuzzy Optimization and Decision Making*, *7*, 257–267.

SESAR Joint Undertaking. (2010, February). *SESAR Factsheet.* (Available Online)

Sethian, J. A. (2010). *Ordered Upwind Methods.* Available from `https://math.berkeley.edu/~sethian/2006/Explanations/ordered_upwind_explain.html` (Accessed 1 July 2016)

Sethian, J. A., & Vladimirsky, A. (2003). Ordered Upwind Methods for Static Hamilton-Jacobi Equations: Theory and Algorithms. *Society for Industrial and Applied Mathematics*, *41*(1), 325–363.

Uçan, F., & Altilar, D. (2012). Using Genetic Algorithms for Navigation Planning in Dynamic Environments. *Applied Computational Intelligence and Soft Computing.* (Article ID 560184)

Upadhyay, S., & Ratnoo, A. (2016). Smooth Trajectory Planning for MAVs with Airspace Restrictions. In *AIAA Guidance, Navigation, and Control Conference.* San Diego: AIAA SciTech.

U.S.-EU MOC Annex 1 - Coordination Committee. (2014). *NextGen/SESAR State of Harmonisation Document.*

Wynnyk, C., Balakrishna, M., Macwilliams, P., & Becher, T. (2013). 2011 Trajectory Based

Operations Flight Trials. In *Tenth USA/Europe Air Traffic Management Research and Development Seminar.* Chicago.

Yang, Z., Fang, Z., & Li, P. (2016). Bio-inspired Collision-free 4D Trajectory Generation for UAVs Using Tau Strategy. *Journal of Bionic Engineering*, *13*(1), 84–97.