Transportation Research Part C 129 (2021) 103232

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Informed scenario-based RRT* for aircraft trajectory planning under ensemble forecasting of thunderstorms

Eduardo Andrés [a,*], Daniel González-Arribas [a], Manuel Soler [a], Maryam Kamgarpour [b], Manuel Sanjurjo-Rivo [a]

[a] Department of Bioengineering and Aerospace Engineering, Universidad Carlos III, Madrid, Spain
[b] Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada

ARTICLE INFO

Keywords:
Aircraft path planning
Sampling-based algorithms
Uncertain thunderstorm avoidance
Parallel programming

ABSTRACT

Thunderstorms represent a major hazard for flights, as they compromise the safety of both the airframe and the passengers. To address trajectory planning under thunderstorms, three variants of the scenario-based rapidly exploring random trees (SB-RRTs) are proposed. During an iterative process, the so-called SB-RRT, the SB-RRT* and the Informed SB-RRT* find safe trajectories by meeting a user-defined safety threshold. Additionally, the last two techniques converge to solutions of minimum flight length. Through parallelization on graphical processing units the required computational times are reduced substantially to become compatible with near real-time operation. The proposed methods are tested considering a kinematic model of an aircraft flying between two waypoints at constant flight level and airspeed; the test scenario is based on a realistic weather forecast and assumed to be described by an ensemble of equally likely members. Lastly, the influence of the number of scenarios, safety margin and iterations on the results is analyzed. Results show that the SB-RRTs are able to find safe and, in two of the algorithms, close-to-optimum solutions.

1. Introduction

Uncertainties inherent to convective weather constitute a major challenge for the Air Traffic Management (ATM) system, affecting its safety, capacity and efficiency, and accounting for a quarter of the en-route delays in Europe (Eurocontrol, 2020). Specifically, thunderstorms represent an important threat, as they involve phenomena such as strong turbulence, wind shear or hail. It is essential to avoid them to ensure both passenger's comfort and aircraft's structural integrity. Thunderstorms' location and timing are hard to predict with certainty. This stochasticity is an important element that methodologies for aircraft trajectory planning must take into account. The aim of this paper is to design a trajectory planning technique for aircraft flying in areas with uncertain thunderstorm development. The objective of the algorithm is to minimize the distance flown to reach a desired target while guaranteeing safety within a user-defined threshold.

The aforementioned problem can be formulated as a stochastic trajectory optimization. Within this class of problems, the motion of aerial vehicles considering an uncertain environment has been addressed in the literature with a wide spectrum of approaches (for a good survey of the topic, see (Dadkhah and Mettler, 2012)). Nevertheless, while there exist multiple methodologies in the field, only

* Corresponding author.
E-mail address: eandres@ing.uc3m.es (E. Andrés).

https://doi.org/10.1016/j.trc.2021.103232
Received 4 December 2020; Received in revised form 11 May 2021; Accepted 18 May 2021
Available online 2 July 2021

few works have tackled aircraft trajectory in stochastic stormy regions. In Seenivasan et al. (2020), nonlinear model predictive control is used to avoid moving storms modelled as ellipsoids whose position and size are changing. Although the problem is solved in a deterministic framework, the authors suggest how to extend the formulation and include uncertainties. In González-Arribas et al. (2019b), a Robust Optimal Control (ROC) problem considering aircraft dynamics and uncertain evolution of convective weather is solved using nonlinear programming. To overcome the sensitivity to the initial guess, a randomized initialization is proposed, to obtain different local optima, among which the best solution is selected. Moreover, ROC is used in Soler et al. (2020) to obtain efficient trajectories based on indicators of risk of convection. Alternatively, Dynamic Programming (DP) algorithms, such as the so-called stochastic Reach-Avoid, can find the optimal trajectory in uncertain and dynamic scenarios, as the authors showed in Hentzen et al. (2018) with application to thunderstorm encounters. However, DP methodologies involve discretization and exploration of the entire state space, a process that is usually computationally prohibitive. Consequently, due to the curse of dimensionality, the affordable dimension of the problem is often limited to 3 states under the discretization approach. Both works, González-Arribas et al. (2019b) and Hentzen et al. (2018), rely on optimal control principles. Although these methods can be used with relatively complex, nonlinear dynamical systems (e.g., an aircraft 3 degrees-of-freedom model in González-Arribas et al. (2019b)), their flexibility to incorporate operational constraints is rather limited.

Contrary to optimal control based methods, path planning and meta-heuristic algorithms look within a set of feasible solutions to find the one closer to the optimum. Within the first class of approaches, we find graph based methods, such as A*, D* or Dijsktra's shortest path (DSP) algorithm. As an example, an A* search is used in Chen et al. (2012) to avoid regions of convective activity near airports. The authors in Taylor et al. (2018), Ng et al. (2009) employ variations of DSP to generate a set of reroutes based on weather avoidance fields (see Section 2). In Bhattacharya et al. (2015), the problem is addressed with persistent homology, looking for the group of trajectories that maximizes the probability of success. However, it requires discretization of the search space and computational time increases quadratically, leading to simulations of several minutes, which is not compatible with real time flight simulation. Alternatively, meta-heuristic algorithms perform different operations on the candidate solutions to improve them. These kinds of operations are usually random and based on natural phenomena such as genetics or animal behaviour. For a good survey on the topic, see (Blum and Roli, 2001). There exist many different meta-heuristic algorithms that have been used for aircraft motion planning considering uncertainties. In particular, in Courchelle et al. (2019), simulated annealing is proposed to solve aircraft conflicts assuming that wind and temperature are not deterministic. Moreover, in González-Arribas et al. (2019a), the authors solve a multi-objective optimization of fuel burn, time of flight and spread in arrival times using a genetic algorithm. In the problem, wind is assumed to be the only source of uncertainty. One common element from Courchelle et al. (2019) and González-Arribas et al. (2019a) is the use of ensemble-based weather products to characterize uncertainties. Additionally, a particle swarm optimization is applied in Hong et al. (2019) to the problem of aircraft sequencing and scheduling subject to uncertain parameters.

A combination of both a meta-heuristic and path planning algorithm was introduced as the so-called Rapidly Exploring Random Tree (RRT) (LaValle, 1998). RRT methods have demonstrated the ability to find safe trajectories between two states in high dimensional problems considering system dynamics and constraints, such as velocity limitations, obstacles or other vehicles (LaValle and Kuffner, 2001). A few years later, the RRT* (Karaman and Frazzoli, 2011) was presented, an update that ensures both feasibility and, unlike the RRT, optimality of the solution. RRT* does not require an initial guess, as it finds a feasible path in a few iterations and then optimizes it in subsequent steps. RRT and RRT* are versatile algorithms with a wide variety of applications in several engineering fields. From their early days, they have been used mainly in robotics (see, for example, (Moon and Chung, 2015; Ghosh et al., 2019)), and their implementation in autonomous driving cars (Kuwata et al., 2009), UAV flights (Pharpatara et al., 2017) or medicine (Duindam et al., 2010) has been explored recently. RRT algorithms can be modified and upgraded by including additional heuristics that enhance performance. A first variation addresses applications in which the environment is time-varying. In these dynamic cases, online RRTs must be considered. Such algorithms are constantly incorporating new data from the surrounding area and take the necessary actions to ensure safety based on the sensed information (e.g., (Kuwata et al., 2009; Frazzoli et al., 2002)). A second relevant upgrade tackles the growth of RRTs considering uncertainties. In Fulgenzi (2009), the author designed a RRT for navigation in areas of uncertain obstacles. Furthermore, the work in Aoude et al. (2013) added the stochasticity in the dynamics of both linear and nonlinear systems. Note that the latter was built on chance constraints and only checked the safety of discrete states along the trajectory. Lastly, to increase the efficiency of the optimal versions (e.g., RRT*), the Informed RRT* was introduced in Gammell et al. (2017), improving convergence times by means of reducing the search space progressively. The previous works on RRTs have not addressed the particular problem of aircraft trajectory planning under environmental uncertainties. Based on recent advances in Numerical Weather Prediction (NWP), our goal is to build RRT algorithms compatible with Ensemble Prediction Systems (EPS). The EPS characterize uncertainties by delivering a series of possible atmospheric realizations.

The contributions of the paper are twofold: First, we apply three RRT-based algorithms to the planning of aircraft trajectories in stochastic weather regions. Second, we provide a method to grow RRTs in areas with uncertainties characterized by ensemble-based products. The present paper fills these two gaps by proposing a set of novel algorithms: the so-coined scenario-based rapidly-exploring random trees (SB-RRTs). The SB-RRT, the SB-RRT* and the Informed SB-RRT* are RRT-based methodologies for aircraft trajectory planning in uncertain environments. They guarantee flight safety in hazardous areas captured by an EPS. In the near future, meteorological products are expected to come as EPS, and this formulation allows the direct use of this data with no processing (e.g.

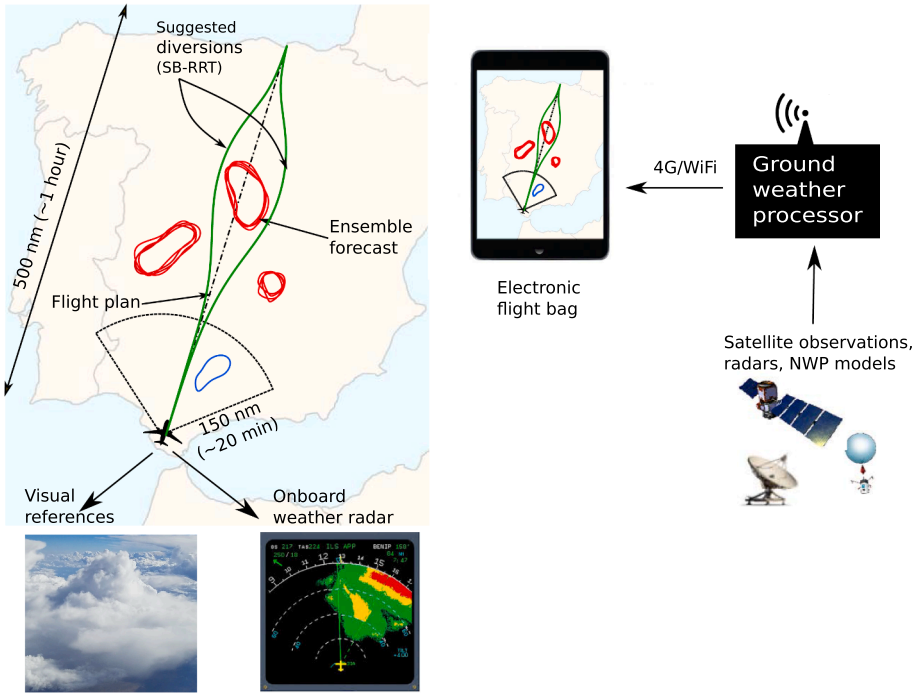


Fig. 1. Representation of a ground-air data link in combination with an onboard electronic flight bag. An ensemble forecast of 4 members provided by the ground weather processor is represented in red. Convective activity detected by the onboard weather radar is shown in blue. Additionally, the flight planning module, the SB-RRT, would suggest possible trajectories to avoid storms (green lines). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

transformation into a unique probability map.). To illustrate our approach, the three proposed algorithms are tested on an aircraft trajectory planning problem utilizing realistic EPS data. A kinematic model of an aircraft flying straight between fixed positions at constant flight level and velocity is considered. Although the algorithms are able to handle more accurate dynamical models, the aim of this work is to illustrate how to incorporate EPS in trajectory planning. To enhance the computational time, the methodology is parallelized with GPU computing techniques leading to a decrease in execution times from days to seconds/minutes.

The paper presents the detailed mathematical formulation for the scenario-based methodology and extends the preliminary results obtained in Andrés et al. (2020) for the RRT*. In addition, the methodology is applied to the RRT and the Informed RRT* as well, comparing computational times and convergence to optimal solutions. The rest of the paper is structured as follows. We motivate our approach in Section 2. The set of SB-RRT algorithms are presented in Section 3. Their performance is evaluated and verified in Section 4. Finally, conclusions are drawn and possible future works are outlined in Section 5.

2. Problem framework

Thunderstorms are considered to be one of the most hazardous phenomena by aviation stakeholders, and it is preferred to avoid them whenever possible. Nevertheless, it is difficult for pilots to find the safest route in stormy regions. This results in delayed and diverted flights as well as in an increase in fuel consumption and total costs.

A first challenge in addressing thunderstorms is that they are uncertain phenomena. Their evolution happens in short timescales ($\sim$ 30 min), which makes them hard to predict. Secondly, pilots use weather charts for flight planning that become outdated as they are not updated during flights; this obsolescence is more significant for long haul routes. Any relevant weather changes are communicated by the air traffic controller or other aircraft. These charts are obtained through NWP models some days in advance and lack the sufficient time and space resolution to capture convective phenomena (such as thunderstorm birth and growth). There exist additional low resolution products that cover wide regions, such as Convective Significant Meteorological Information (SIGMET), used by pilots to determine no-fly regions. Eventually, during a flight, the main source of recent weather information is the onboard weather radar. These systems present multiple constraints as their range is limited to 150 nm (around 20 min of flight) and they are only able to scan

the aircraft front region. With the radar alone there is no information about lateral areas, which might be relevant for possible diversions.

In this context, there exist many ongoing efforts to improve the weather data available for aviation. In the first place, the atmosphere is a nonlinear and chaotic system and there might exist huge differences between predictions. For this reason, weather analysis should not be based on just one forecast but on an EPS. These products deliver between 10 and 50 numerical weather forecasts assuming small perturbations in the models and the parameters (Bouttier et al., 2012; WMO, 2012). It must be noted that existing EPS are not yet able to capture the phenomenology behind convective events. Nonetheless, in the near future (5–10 years), NWP methods are expected to evolve towards very short term (∼1 h) and very high resolution (∼100 m) convective-permitting EPS able to better capture thunderstorms (Bauer et al., 2015). As a consequence, future algorithms for trajectory planning must be able to work with ensemble-based weather products that would eventually be able to forecast thunderstorms.

A second effort is focused on the ground-air uplink of data and the fusion with onboard information. To this end, NWP forecasts are combined with radar, satellite and other possible observations and displayed on the cockpit (Forster et al., 2016). Although this approach has been successfully tested in past projects such as FLYSAFE (Lunnon et al., 2009) or eFlightOps (Kessinger et al., 2015), its actual deployment in commercial aircraft is still under research. Aviation is subject to strict regulation and certification processes that need to be overcome before these systems are ready to be included in primary flight displays. However, the use of complementary devices (electronic flight bags, tablets) to represent additional weather information would provide pilots with more time to react to thunderstorm evolution. As a result, more efficient trajectory diversions and fuel savings could be achieved. An example of that type of technology is eWAS Pilot,[1] an app that provides pilots with real time weather information from several sources through Wifi or 4G connections.

The aim of the SB-RRTs is to potentially be used in the cockpit together with the aforementioned systems. Using the information onboard, the algorithms would suggest to the pilot possible detours from the initial flight plan to avoid any dangerous weather event and minimize costs. Nevertheless, the use of this tool would not be limited to the pilot, yet, in congested airspaces could be used by ATCOs to coordinate with other flights and decide the best possible diversions around storms. A sketch of the proposed solution is represented in Fig. 1. Tactical rerouting is a topic of interest and has been widely explored and tested in the last decade. A case in point is Airspace Technology Demonstration 3 (ATD-3) (Sheth et al., 2018), a project by NASA that addresses the improvement of cruise and arrival phases. ATCOs are usually under a heavy workload and rely on conservative rules to quickly plan routes around thunderstorms that might not be efficient. Hence, ATD-3 aims to provide pilots with small route changes that consider winds, convective weather, airspace restrictions and other aircraft to minimize time and fuel consumption. Tools such as TASAR (Henderson, 2013) and DWR (McNally et al., 2015) tackle cruise phase while DRAW (Isaacson and Gong, 2018) and DAR (Gong and McNally, 2015) focus on the arrival stage. All of them are continuously identifying flights that can benefit from rerouting, updating possible trajectories every 12 s. Conversely, the work in Ng et al. (2009) proposes a methodology based on dynamic programming to obtain reroutes that are less likely to be modified by weather. In Taylor and Wanke (2012), the authors designed a simulated annealing algorithm to provide different acceptable diversions considering weather, congestion or flight distance. An alternative class of methods is based on Trajectory Option Sets (TOS), a group of possible alternative routes for flights (before departure or airborne). In Evans and Lee (2019), machine learning techniques are used to automatically build a TOS based on historical data. Additionally, the authors in Taylor et al. (2018) address the problem with a multi-objective genetic algorithm that, based on different metrics (e.g., incursions in adverse weather regions), evaluates the acceptability of each member in a TOS.

One common aspect from these methodologies is the modeling of convective weather with Weather Avoidance Fields (WAF) (Matthews and Delaura, 2010), a series of deterministic polygons that represent regions that pilots would often try to avoid. For example, a 70 percent WAF represents weather regions that 70% of the pilots are likely to avoid (this is the value typically considered in these works). In the present work, an alternative methodology for tactical rerouting is suggested, able to run in similar times (e.g., the Informed SB-RRT* provides optimal paths in $\sim$ 10 seconds). Instead of WAF, the proposed algorithms work with EPS, a trend in meteorology that in the near future will provide high resolution weather forecasts and characterization of uncertainties (Bauer et al., 2015). Depending on the scope of the simulation, the SB-RRTs can provide trajectories between close waypoints or for the entire cruise phase. Moreover, by means of a user-defined safety margin, there is a compromise between reducing flight distance and increasing safety. Note that although the algorithm focuses on weather events, it can be extended to consider congested sectors or restricted regions.

3. Scenario-based RRTs

This section presents the set of Scenario-Based RRTs: SB-RRT, SB-RRT* and Informed SB-RRT*. The methodology to handle unsafe sets described by a finite number of scenarios is included.

[1] http://www.ewas.aero/product/ewas-pilot.

3.1. Scenario-based environment

Let $X \subset \mathbb{R}^{d_x}$ be the space of dimension d_x in which both the aircraft flight and the collision avoidance take place. The space X only includes possible aircraft coordinates, hence $d_x = 2$ or 3. Let $X_{unsafe} \subset X$ be the unsafe set that must be avoided. The complementary set $X_{safe} = X \backslash X_{unsafe}$ represents the safe-to-fly regions. In this formulation, the set X_{unsafe} represents an ensemble forecast of thunderstorms with the total number of possible scenarios or members denoted by N_{sc}. A member of the ensemble is formed by several storm cells, each of which is denoted by O_l^j. Superscript j, with $j = 1,\ldots,N_{sc}$, refers to a particular member of the ensemble, while subscript l, with $l = 1,\ldots,N_o^j$, considers a storm cell from the j-th member. The number of storm cells N_o^j, in general, varies from scenario to scenario.

A storm cell O_l^j is a closed region delimited by a general polhyedron. Each cell is treated as deterministic, and it can be checked whether a state configuration lies inside it or a curve intersects it. Let X_{unsafe}^j be the j-th member of the ensemble and the union of all the storm cells from that member. Then,

$$X_{unsafe}^j = \bigcup_{l=1}^{N_o^j} O_l^j. \tag{1}$$

The ensemble X_{unsafe} is the set that includes the different members given by X_{unsafe}^j, hence

$$X_{unsafe} = \{X_{unsafe}^1, \ldots, X_{unsafe}^{N_{sc}}\}. \tag{2}$$

Ensemble based weather products produce equiprobable ensemble members by default, as there is no way of knowing if any of them would be more important beforehand. In consequence, in this paper, each of the members of the ensemble is assumed to occur with identical probability. That is $\Pr(X_{unsafe} = X_{unsafe}^j) = 1/N_{sc}, \forall j$. The risk of a particular region can be estimated by accounting for the proportion of members of the ensemble that predict a storm.

3.2. SB-RRT algorithm

Let $S \subset \mathbb{R}^{d_s}$, with $d_s \geq d_x$, be the state space[2] and let $U \subset \mathbb{R}^{d_u}$ be the control space. The aircraft dynamics is represented by a state vector $\mathbf{s} \in S$ that evolves according to a transition equation,

$$\dot{\mathbf{s}} = f(\mathbf{s}, \mathbf{u}), \tag{3}$$

where $\mathbf{u} \in U$ is the control input.

The SB-RRT algorithm is an iterative process to grow a tree $\mathcal{T} \subset X$, formed by nodes and edges, given a maximum number of iterations *MaxIter*. The aim of $\mathcal{T}$ is to explore the free space X_{safe} and find a safe path between a pair of state configurations. Let the node a be the representation in X of a state $\mathbf{s} \in S$ obtained randomly. Moreover, each node can be considered as a programming object with a spatial representation in X and multiple fields associated to relevant variables (e.g., velocity, heading, distance to the origin.). Let the edge e represent the propagation of the dynamics given by Eq. (3) between two states (or nodes) $\mathbf{s}$ and $\mathbf{s}'$. The tree $\mathcal{T} = (\mathcal{A}, \mathcal{E})$ consists of two sets, $\mathcal{A}$ and $\mathcal{E}$, populated by nodes and edges respectively. The condition to be included in $\mathcal{A}$ or $\mathcal{E}$ is based on the safety requirements, which leads to the rejection of many nodes and edges through the iterations.

Let $\mathbf{s}_{start}, \mathbf{s}_{goal} \in S$ be the initial and goal state configurations. The nodes a_{start} and a_{goal} are associated to these states.[3] The way the SB-RRT grows is analogous to the RRT (LaValle, 1998), with the main difference being the way safety is ensured. In particular, during its growth, the tree allocates risk dynamically to an edge so as to avoid the violation of a user-defined safety margin. The following list of functions is required:

- *InitializeRRT*: it creates the tree set $\mathcal{T}$ with two empty sets, the node set $\mathcal{A}$ and the edge set $\mathcal{E}$.
- *RandomSample*: it takes a random state $\mathbf{s}_k \in S$ and creates the node a_k. Subscript k indicates that the sample is obtained during the k-th iteration.
- *NearestNode*: it returns the closest node to the sample $a_k, a_{nearest} \in \mathcal{A}$, according to a predefined metric, e.g., Euclidean distance or Dubins path length.
- *Steer*: it drives the system from a node a to another a' minimizing a cost function, e.g. distance, time or fuel consumption. This connection between nodes is an edge e.

[2] The state space corresponding to the dynamics of the aircraft is denoted by S, while X represents the space where collision avoidance happens (see Section 3.1). The space X is limited to possible aircraft positions whereas S can also include variables such as velocity or heading angle.

[3] Although a_{start} and a_{goal} are created beforehand, they are only included in $\mathcal{A}$ once *AddNode* function is called. This applies to any other node a.

- *Safe*: it checks the safety of any edge e before being included in $\mathcal{E}$. The way safety is evaluated, and what differentiates the SB-RRT from a traditional RRT, is a contribution of this paper. See SubSection 3.5.
- *Cost*: it calculates the cost of the path between two nodes a and a'.
- *Parent*: for any node $a \in \mathcal{A}$ it returns its parent node (the previous node to which is connected) $a_{parent} \in \mathcal{A}$ and the edge $e_{parent} \in \mathcal{E}$ connecting both. The node a is called child of a_{parent}
- *AddNode*, *AddEdge*: these functions include a node in $\mathcal{A}$ or an edge in $\mathcal{E}$, respectively.
- *RemoveNode*, *RemoveEdge*: these functions remove a node from $\mathcal{A}$ or an edge from $\mathcal{E}$, respectively.

Algorithm 1
$\mathcal{T} = (\mathcal{A}, \mathcal{E}) \leftarrow$ SB-RRT(a_{start}, a_{goal})

```
1: T ← InitializeRRT();
2: A ← AddNode(a_start);
3: while k < MaxIter do
4:    a_k ← RandomSample();
5:    a_nearest ← NearestNode(a_k, A);
6:    e_nearest ← Steer(a_nearest, a_k);
7:    if Safe(e_nearest) then
8:       A ← AddNode(a_k);
9:       E ← AddEdge(e_nearest);
10:   end if
11:   if a_goal ∈ A [4] then
12:      stop
13:   end if
14: end while
15: return T
```

The SB-RRT pseudocode is summarized in Algorithm 1. During each iteration, the algorithm takes a random state $\mathbf{s}_k$, creates the corresponding node a_k and connects it to the closest node $a_{nearest}$ that is already in $\mathcal{T}$. If the connection is not feasible (e.g., not safe, collision with obstacles) and *Safe* fails, a_k is rejected and the iterative process continues with a new sample. After *MaxIter* iterations, the algorithm returns tree $\mathcal{T}$ made of safe paths to each node in $\mathcal{A}$.

Note that one of the main goals of the SB-RRT (and subsequent RRT-based algorithms) is to include the node a_{goal} in the set $\mathcal{A}$ so that it is related to a_{start}. Parent-child relations are an important part of the algorithm; one node can have multiple children, but each node only has one parent. Therefore, a_{start} can be reached by choosing an arbitrary node from $\mathcal{A}$ and tracking the sequence of parents. In consequence, once a_{goal} is a part of the tree,[4] a safe trajectory towards a_{start} is found. By the time the SB-RRT finds a safe trajectory, the algorithm is not able to improve it as there is no reorganization of the tree, and the iterative process can stop (Algorithm 1, line 12).

3.3. SB-RRT* algorithm

The SB-RRT presented in the previous section only finds feasible trajectories but performs no optimization. In this section, the SB-RRT* is presented. It is a modification of the asymptotically optimal RRT* algorithm (Karaman and Frazzoli, 2011) with the scenario-based heuristic for checking safety. To achieve the optimization, three additional functions are required:

- *Near*: it gets the set of nodes $A_{near} \subseteq \mathcal{A}$ within a ball centered at a_k with radius $\mu > (\frac{\log card(\mathcal{A})}{card(\mathcal{A})})^{\frac{1}{d_x+1}}$ (see Solovey et al., 2020), where μ is a constant to ensure optimality and $card(\mathcal{A})$ is the cardinality of $\mathcal{A}$ during the corresponding iteration.
- *BestParent*: among the set A_{near}, it finds the node $a_{parent} \in \mathcal{A}$ that achieves the smallest cost going from $a_{start} \in \mathcal{A}$ to a_k passing by a_{parent}. The node a_{parent} is chosen as the parent of a_k, and the connection between both is the edge e_{parent}. Note that if none of the explored connections is safe, then the parent node is $a_{nearest}$, with the corresponding connection $e_{nearest}$ which is already safe. See Algorithm 3.
- *Rewire*: it checks if the cost of going from $a_{start} \in \mathcal{A}$ to each of the elements in A_{near} can be reduced by going through a_k. The rewiring process eliminates existing edges and creates new ones between nodes, changing the relations between parent and child nodes. See Algorithm 4.

[4] The probability of sampling exactly a_{goal} during any iteration is 0. This can be addressed by defining a ball centered in a_{goal} and looking for samples within it. Alternatively, an attempt to connect a_{goal} to the tree can be made during an iteration chosen at random. In this paper, the second methodology is implemented as it reduced the number of iterations required to connect a_{goal}. In particular, let b be a fixed number between 0 and 1 ($b = 0.1$ in our simulations) and let $r \sim \mathcal{U}(0,1)$ (with $\mathcal{U}$ a uniform distribution). If $r \leqslant b$, the new sample is equal to a_{goal}, and random otherwise.

Algorithm 2

$\mathcal{T} = (\mathcal{A}, \mathcal{E}) \leftarrow$ SB-RRT*(a_{start})

$\mathcal{T} \leftarrow InitializeRRT()$;
$\mathcal{A} \leftarrow AddNode(a_{start})$;
while $k < MaxIter$ **do**
  $a_k \leftarrow RandomSample()$;
  $a_{nearest} \leftarrow NearestNode(a_k, \mathcal{A})$;
  $e_{nearest} \leftarrow Steer(a_{nearest}, a_k)$;
  if $Safe(e_{nearest})$ **then**
    $A_{near} \leftarrow Near(a_k, \mathcal{A})$;
    $a_{parent}, e_{parent} \leftarrow BestParent(a_k, a_{nearest}, e_{nearest}, A_{near})$;
    $\mathcal{A} \leftarrow AddNode(a_k)$;
    $\mathcal{E} \leftarrow AddEdge(e_{parent})$;
    $\mathcal{T} \leftarrow Rewire(a_k, A_{near})$;
  end if
end while
return $\mathcal{T}$

Algorithm 3

$a_{parent}, e_{parent} \leftarrow BestParent(a_k, a_{nearest}, e_{nearest}, A_{near})$

$a_{parent} \leftarrow a_{nearest}$; $e_{parent} \leftarrow e_{nearest}$;
$c_{min} \leftarrow Cost(a_{start}, a_{nearest}) + Cost(a_{nearest}, a_k)$;
for $a_{near} \in A_{near}$ **do**
  $e_{near} \leftarrow Steer(a_{near}, a_k)$;
  if $Safe(e_{near})$ **then**
    $c_{near} \leftarrow Cost(a_{start}, a_{near}) + Cost(a_{near}, a_k)$;
    if $c_{near} < c_{min}$ **then**
      $a_{parent} \leftarrow a_{near}$; $e_{parent} \leftarrow e_{near}$;
      $c_{min} \leftarrow c_{near}$;
    end if
  end if
end for
return a_{parent}, e_{parent}

Algorithm 4

$\mathcal{T} \leftarrow Rewire(a_k, A_{near})$

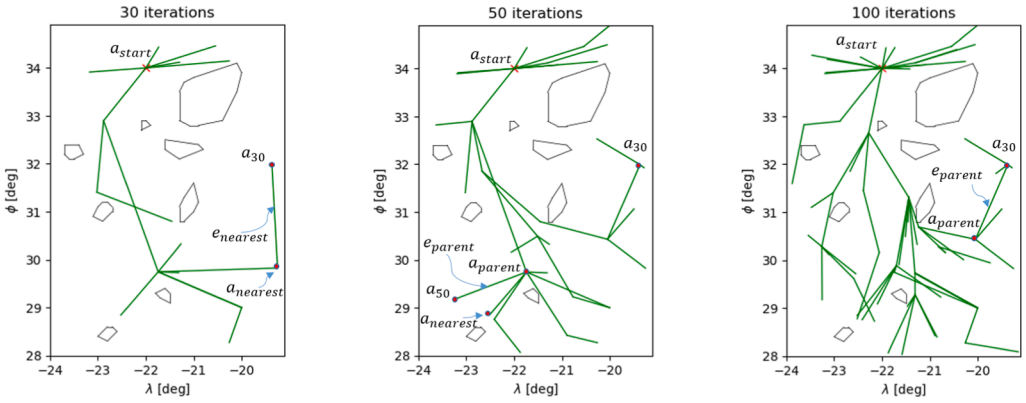
for $a_{near} \in A_{near}$ **do**
  $e_{near} \leftarrow Steer(a_k, a_{near})$;
  $c_{near} \leftarrow Cost(a_{start}, a_k) + Cost(a_k, a_{near})$;
  if $Safe(e_{near})$ **then**
    if $c_{near} < Cost(a_{start}, a_{near})$ **then**
      $a_{parent}, e_{parent} \leftarrow Parent\ (a_{near})$;
      $\mathcal{E} \leftarrow RemoveEdge(e_{parent})$;
      $\mathcal{E} \leftarrow AddEdge(e_{near})$;
    end if
  end if
end for
return $\mathcal{T}$

The SB-RRT* pseudocode is summarized in Algorithm 2. Additionally, *BestParent* and *Rewire* functions are included in Algorithms 3 and 4. In each iteration, the algorithm takes a random sample a_k and looks for the path of minimum cost towards the origin a_{start}. Moreover, it optimizes the connections of the nodes surrounding a_k. The result, after *MaxIter* iterations, is a tree $\mathcal{T}$ that minimizes the cost from a_{start} to each node in $\mathcal{A}$.

The original RRT* presents asymptotic optimality (Karaman and Frazzoli, 2011) and requires that $MaxIter \rightarrow \infty$ to reach the trajectory of minimum cost. However, running the simulation indefinitely is impossible and *MaxIter* must be large enough to obtain close-to-optimum solutions, which can be post-processed a posteriori. As the SB-RRT* is based on the RRT*, it also requires enough iterations for the optimization routines, *BestParent* and *Rewire*, to be effective. Typically, with a higher number of obstacles, more iterations are required.

An example of how the SB-RRT* grows is shown in Fig. 2, illustrating the process during three different iterations. It can be observed that:

- In Fig. 2(a), the sample taken during the 30-th iteration, a_{30}, is connected to the closest node $a_{nearest} \in \mathcal{A}$.



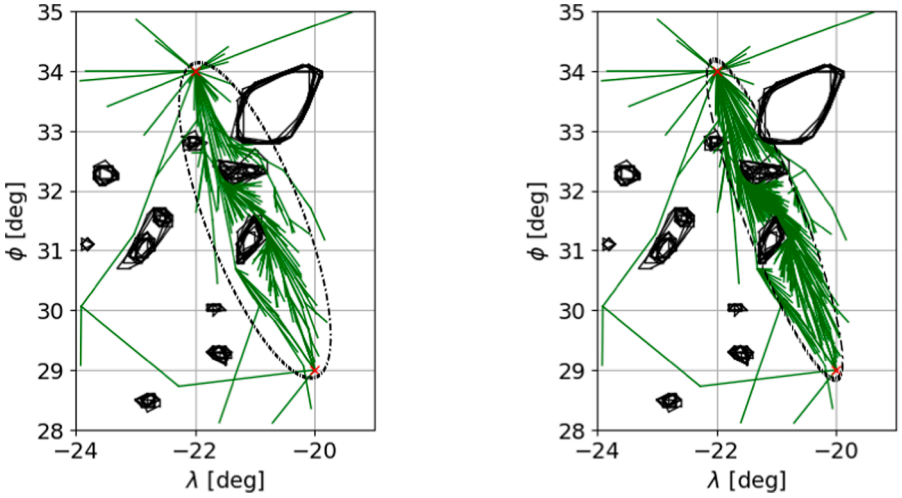
(a) 30 iterations. (b) 50 iterations. (c) 100 iterations.

Fig. 2. Example of SB-RRT* evolution with 30, 50 and 100 iterations.

- In Fig. 2(b), the 50-th sample, a_{50}, is connected to $a_{parent} \in \mathcal{A}$, despite the fact that $a_{nearest} \in \mathcal{A}$ is the closest node. This is a consequence of the *BestParent* function. At first, a_{50} would be connected to $a_{nearest}$, but there is one $a_{near} \in A_{near}$ that leads to a shortest path to a_{start}.
- In Figs. 2(b) and (c), it is shown that a_{30}, previously connected to its closest node, is now linked to another parent, $a_{parent} \in \mathcal{A}$. In addition, the path between a_{start} and a_{30} in Fig. 2(a) has changed in Fig. 2(c), minimizing the flight distance as the iterations increase. This is a direct effect of the *Rewire* process. For some node a_k, with $k > 30$, a_{30} fell inside A_{near} and it was found that the path to a_{30} could be shortened through a_k.

3.4. Informed SB-RRT* algorithm

One of the main drawbacks of the SB-RRT* (inherited from the RRT*) is its inefficiency in single query path planning problems in which the objective is to find the optimal trajectory between origin and destination. The RRT* does not use the known information about the goal state until the iterative process concludes and needs a large computational effort for the minimization of the costs towards each random state. To tackle this issue, the Informed RRT* methodology was developed (Gammell et al., 2017), which is able to maintain optimality and improve convergence rates by reducing the search space X progressively. The Informed RRT* grows similarly to the RRT* until a solution is found. Then, the sampling region is reduced to an ellipsoidal domain, characterized by the distance from the origin-destination and the cost of the best solution c_{best}. This ensures that only samples that can improve the best solution are considered. Any sample outside the ellipsoid would lead to trajectories with a cost higher than c_{best}, thus obstructing the optimization process. Note that this heuristic is only appropriate for the trajectory planning problem in which the objective is to



(a) 500 iterations. (b) 1000 iterations.

Fig. 3. Evolution of the ellipsoidal sampling region.

minimize distance at constant velocity. This results in a loss of generality with respect to the original SB-RRT* in exchange for the enhancement of convergence rates. For more complex problems, the use of alternative heuristics should be explored.

A sketch of the ellipsoidal domain after 500 and 1000 iterations is shown in Fig. 3. Each time a better solution is found, the ellipsoid shrinks, and, if there were no obstacles, it would eventually collapse into a straight line. The authors recommend referring to Gammell et al. (2017) for further details on this methodology. Again, the Informed SB-RRT* included in Algorithm 5 and presented in this work updates the Informed RRT* with the new safety check procedure. Two additional functions are required:

- *UpdateBestCost*: it returns the cost c_{best} of the current solution between a_{start} and a_{goal}. This value will be changing due to *Rewire* function.
- *InformedSample*: it returns a random sample within an ellipsoidal region with foci on a_{start} and a_{goal}, minor axis equal to the Euclidean distance between both and major axis equal to c_{best}.

Algorithm 5
$\mathcal{T} = (\mathcal{A}, \mathcal{E}) \leftarrow$ Informed SB-RRT*(a_{start}, a_{goal})

1: $\mathcal{T} \leftarrow$ *InitializeRRT*();
2: $\mathcal{A} \leftarrow$ *AddNode*(a_{start});
3: $c_{best} \leftarrow \infty$;
4: **while** $k <$ *MaxIter* **do**
5: $c_{best} \leftarrow$ *UpdateBestCost*$(\mathcal{A})$;
6: **if** $c_{best} < \infty$ **then**
7: $a_k \leftarrow$ *InformedSample*$(a_{start}, a_{goal}, c_{best})$;
8: **else**
9: $a_k \leftarrow$ *RandomSample*();
10: **end if**
11: $a_{nearest} \leftarrow$ *NearestNode*$(a_k, \mathcal{A})$;
12: $e_{nearest} \leftarrow$ *Steer*$(a_{nearest}, a_k)$;
13: **if** *Safe*$(e_{nearest})$ **then**
14: $A_{near} \leftarrow$ *Near*$(a_k, \mathcal{A})$;
15: $a_{parent}, e_{parent} \leftarrow$ *BestParent*$(a_k, a_{nearest}, e_{nearest}, A_{near})$;
16: $\mathcal{A} \leftarrow$ *AddNode*(a_k);
17: $\mathcal{E} \leftarrow$ *AddEdge*(e_{parent});
18: $\mathcal{T} \leftarrow$ *Rewire*(a_k, A_{near});
19: **end if**
20: **end while**
21: **return** $\mathcal{T}$

3.5. Evaluation of safety

In Algorithm 1 (line 7), Algorithm 2 (line 7), Algorithm 3 (line 5), Algorithm 4 (line 4) and Algorithm 5 (line 13) the function *Safe* is called to determine if an edge is safe and suitable to be added to the tree. Assuming deterministic unsafe regions, an edge would be safe if there is no intersection with any hazardous area. However, if the environment is uncertain, it is only known up to a certain probability if the edge is safe. The heuristic to accept or reject an edge as a part of the tree are presented below and are valid for any of the SB-RRTs presented above. By applying them, any possible sequence of edges in $\mathcal{T}$ will be safe with a user-defined probability.

Probabilistic safety of an edge. Before describing the safety of the approach, we describe how we incorporate the ensemble forecast data in a probabilistic safety evaluation. In particular, we define the probability of any edge e^i being unsafe as the ratio of the number of scenarios in which the edge intersects X_{unsafe} and the total number N_{sc},

$$\Pr(e^i \cap X_{unsafe} \neq \varnothing) = \frac{m^i}{N_{sc}}, \tag{4}$$

where m^i is equal to the number of scenarios where an intersection takes place and is defined as,

$$m^i = \sum_{j=1}^{N_{sc}} m^{i,j}, \tag{5}$$

with

$$m^{i,j} = \begin{cases} 0 & \text{if } e^i \cap X^j_{unsafe} = \varnothing, \\ 1 & \text{otherwise.} \end{cases} \tag{6}$$

The variable $m^{i,j}$ indicates if there is an intersection between an edge e^i and a realization of the unsafe set, X^j_{unsafe}, as defined in Section

3.1.

Safety of the tree. Henceforth, the k-th iteration is considered, with $k \leqslant MaxIter$. Let $\epsilon \in [0,1]$ be the maximum admissible risk for any possible trajectory. The existing tree, created during the previous $k-1$ iterations, is assumed to be safe with a margin ϵ, and the safety of any new edge is built on this fact. According to Algorithms 1–5, there are only three possible types of trajectories that appear during each iteration and involve both a_{start} and a_k:

- First, the trajectory connecting a_{start} to a_k through $a_{nearest}$ using the edge $e_{nearest}$.
- Second, the trajectories created with *BestParent* function, between a_{start} and a_k through each a_{near} using the edge e_{near} (only for SB-RRT* and Informed SB-RRT*).
- Third, any of the rearrangements that happen with *Rewire* function, connecting a_{start} to a_{near}, and going through a_k with the edge e_{near} (only for SB-RRT* and Informed SB-RRT*).

Although in the following we verify safety of the approach for the first kind of trajectories, its generalization to the remaining two kinds is immediate. Let $P_k = \{e^1, \ldots, e^{N_k}\}$ be the trajectory between a_{start} and a_k via $a_{nearest}$. The set P_k represents a sequence of edges e^i, with $i = 1,\ldots,N_k$. Note that e^{N_k} represents the edge connecting the last two nodes whose safety needs to be checked, which in this case is $e_{nearest}$. The trajectory P_k is considered to be safe, with a safety margin ϵ, if all its elements are safe (Lefkopoulos and Kamgarpour, 2019). That is,

$$\Pr(P_k \in X_{safe}) = \Pr\left(\bigwedge_{i=1}^{N_k} e^i \in X_{safe}\right) \geqslant 1-\epsilon. \tag{7}$$

Eq. (7) is reformulated using De Morgan's law. For a series of events, denoted by Z_i, it states that $\neg(\wedge_i Z_i) = \vee_i(\neg Z_i)$, where $\neg Z_i$ is the complement of event Z_i. If $Z_i = e^i \in X_{safe}$, the complement event corresponds to the existence of intersection between e^i and X_{unsafe}, hence $\neg Z_i = (e^i \cap X_{unsafe} \neq \varnothing)$. Then,

$$\Pr(P_k \cap X_{unsafe} \neq \varnothing) = \Pr\left(\bigvee_{i=1}^{N_k} e^i \cap X_{unsafe} \neq \varnothing\right) \leqslant \epsilon. \tag{8}$$

By means of Boole's inequality, for a finite number of events $Z_i, \Pr(\vee_i Z_i) \leqslant \sum_i \Pr(Z_i)$, and Eq. (8) is conservatively satisfied as follows:

$$\Pr(P_k \cap X_{unsafe} \neq \varnothing) \leqslant \sum_{i=1}^{N_k} \Pr(e^i \cap X_{unsafe} \neq \varnothing) \leqslant \epsilon. \tag{9}$$

We now use the variable m^{ij} defined in (6) to calculate $\Pr(e^i \cap X_{unsafe} \neq \varnothing)$:

$$\sum_{i=1}^{N_k} \Pr(e^i \cap X_{unsafe} \neq \varnothing) = \frac{1}{N_{sc}} \sum_{i=1}^{N_k} \sum_{j=1}^{N_{sc}} m^{i,j} \leqslant \epsilon. \tag{10}$$

Letting $M_k := N_{sc} \sum_{i=1}^{N_k} \Pr(e^i \cap X_{unsafe} \neq \varnothing)$, we obtain the constraint

$$M_k = \sum_{i=1}^{N_k} \sum_{j=1}^{N_{sc}} m^{i,j} \leqslant \lfloor \epsilon N_{sc} \rfloor, \tag{11}$$

where $\lfloor x \rfloor$ denotes the floor function of x. The variable M_k represents the number of times any edge in P_k has intersected any realization of X_{unsafe}. From Eq. (11), it is concluded that the growth of any of the Scenario-Based RRTs must be constrained by limiting the maximum number of intersections with X_{unsafe} to verify Eq. (7). During the growth, the total number of intersections with X_{unsafe} of any trajectory that starts at a_{start} can be of, at most, $\lfloor \epsilon N_{sc} \rfloor$. In order to guarantee safety during the tree growth, this work proposes the *dynamic risk allocation*, a novel method that combines the safety evaluation from Eq. (11) with the random growth of RRT-based algorithms.

Dynamic risk allocation. As it was stated in Section 3.2, one of the advantages of implementing a RRT algorithm is that the nodes can be used to store data such as their position, velocity, the sequence of edges to get to the initial state, the total distance covered or the elapsed time. Using this feature, in our approach we store M_k in the node a_k. As M_k is additive, Eq. (11) can be decomposed into two terms as follows,

$$M_k = \sum_{i=1}^{N_k-1} \sum_{j=1}^{N_{sc}} m^{i,j} + \sum_{j=1}^{N_{sc}} m^{N_k,j} \leqslant \lfloor \epsilon N_{sc} \rfloor, \tag{12}$$

where the first sum corresponds to the value of $M_{nearest}$ associated to $a_{nearest}$, and the second sum corresponds to the number of intersections of the last edge in P_k. Using Eq. (5),

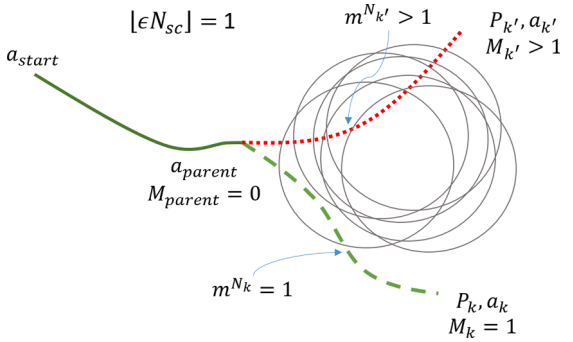


Fig. 4. Schematic illustration of the dynamic risk allocation with two different possibilities: rejection of an edge as it exceeds the maximum number of allowed intersections (red dotted line) and acceptance of an edge that meets the constraints (green dashed line). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$m^{N_k} := \sum_{j=1}^{N_{sc}} m^{N_k j},$$

and Eq. (12) is reformulated as:

$$M_k = M_{nearest} + m^{N_k} \leqslant \lfloor \epsilon N_{sc} \rfloor. \tag{13}$$

Note that during the k-th iteration the only unknown is m^{N_k}, since $M_{nearest}$ was already obtained in a previous iteration. In Algorithm 1, for the edge $e_{nearest}$ to be considered safe, its value of m^{N_k} must satisfy Eq. (13). If $M_k > \lfloor \epsilon N_{sc} \rfloor$, the number of intersections between $e_{nearest}$ and X_{unsafe} is higher than desired. The condition in Eq. (13) can be generalized for any possible trajectory with origin in a_{start}. Let a_k be the final node in the sequence and let $a_{parent} \in \mathcal{A}$ be its parent with a known value of M_{parent}. The connection from a_{parent} to a_k is the edge e_{parent}. Consequently,

$$M_k = M_{parent} + m^{N_k} \leqslant \lfloor \epsilon N_{sc} \rfloor. \tag{14}$$

Eq. (14) is the basis of the dynamic risk allocation proposed in this work, in which the risk is assigned non-uniformly to the edges as the tree is expanding.

As an example, illustrated in Fig. 4, let $\epsilon = 0.1$ and $N_{sc} = 10$. At most, one edge in every possible trajectory can intersect the set X_{unsafe} once, as $\lfloor \epsilon N_{sc} \rfloor = 1$. Let P_k and $P_{k'}$ be two trajectories towards two nodes a_k and $a_{k'}$ respectively. Without loss of generality, let us assume that both nodes share the same parent a_{parent} with a value $M_{parent} = 0$. Then, with Eq. (13),

$$M_k = M_{parent} + m^{N_k} = 1,$$

$$M_{k'} = M_{parent} + m^{N_{k'}} > 1.$$

In this situation, $a_{k'}$ would be rejected, as $M_{k'} > 1$, whereas a_k would be added to the tree. Moreover, the tree can grow from a_k as long as the subsequent edges involve no more intersections. In that way, the tree is able to allocate risk wherever necessary, keeping track of a whole path.

If it is required to know which parts of a trajectory are less safe, the allocated risk for any edge e^i which connects two consecutive nodes $a_{k'}$ and a_k can be measured:

$$\Pr(e^i \cap X_{unsafe} \neq \emptyset) = \frac{m^i}{N_{sc}} = \frac{|M_k - M_{k'}|}{N_{sc}}. \tag{15}$$

If the tree is grown according to the dynamic risk allocation, any trajectory that starts at a_{start} and ends at any node in the tree is safe with a margin ϵ. In particular, the solution towards a_{goal} is safe as well. *Safety of the solution.* Let P^* be the solution of the SB-RRT, the SB-RRT* or the Informed SB-RRT* connecting a_{start} and a_{goal}. At a_{goal}, the constraint $M_{goal} \leqslant \lfloor \epsilon N_{sc} \rfloor$ must be satisfied. From Eq. (9) and Eq. (10),

$$\Pr(P^* \cap X_{unsafe} \neq \emptyset) \leqslant \frac{M_{goal}}{N_{sc}} \leqslant \frac{\lfloor \epsilon N_{sc} \rfloor}{N_{sc}} \leqslant \epsilon. \tag{16}$$

Hence, the safety of the trajectory between a_{start} and a_{goal} is ensured with a user-defined probability $1 - \epsilon$.

Table 1
CPU vs. GPU. Computational time required by *Safe* function with respect to the number of scenarios.

N_{sc}	With CPU (ms)	With GPU (ms)	$\text{time}_{\text{CPU}}/\text{time}_{\text{GPU}}$
5	720	1.22	590
10	1500	1.25	1200
20	2850	1.25	2280
50	6840	1.31	5220

3.6. Parallelized SB-RRTs

The scenario-based formulation requires the implementation of a repetitive task: obtaining intersection between an edge and a series of closed figures. In particular, the application of Eq. (14) (*Safe* function) and the calculation of each m^{N_k} are a potential bottleneck in the SB-RRTs. By using traditional CPU computation the calculation of the intersection of an edge with each ensemble member X^j_{unsafe} is done sequentially. Moreover, as each member usually has several storm cells O^j_l, the intersections with each of them are also determined sequentially. Each time *Safe* is called, the total number of intersections to be obtained is $\sum_{j=1}^{N_{sc}} N^j_o$. In consequence, the computational time required for checking one edge grows linearly with the number of ensemble members and the number of storm cells per member. As the algorithm requires an extensive use of *Safe* function, which increases with the number of nodes, the total execution time grows with the maximum number of iterations.

As it was proposed in Andrés et al. (2020), this issue is addressed with parallelization routines using CUDA (NVIDIA Corporation, 2010), a platform for general computing on GPUs. The result is the set of Parallelized SB-RRTs, in which, to calculate the value of m^{N_k}, all the storm cells O^j_l included in any ensemble member are handled in parallel. In this way, the $\sum_{j=1}^{N_{sc}} N^j_o$ steps required are reduced to 1; *Safe* function is independent of both the number of members in the ensemble and the number of storm cells. In order to compare the performance of the GPU approach, the computational time for each time *Safe* function is called is included in Table 1. A first conclusion is that, using GPU, this time is reduced in 3 orders of magnitude. Moreover, the processing time is nearly independent of the number of ensemble members N_{sc} that are being considered. The computations are performed in a workstation equipped with an Intel Xeon E3-1240 v5 CPU running at 3.5 GHz and a NVIDIA Quadro M4000 GPU of 8 GB.

4. Case study

In this section, the SB-RRTs are tested considering a kinematic model of an aircraft flying with constant heading between nodes. Constant airspeed and flight level are assumed. Note that the algorithms are able to handle more complex dynamical models, but the goal of this work is to demonstrate how to deal with ensemble-based weather products.

The state variables are latitude ϕ, longitude λ and heading angle χ. The planning space $X = [-24^\circ, -19^\circ] \times [28^\circ, 35^\circ]$ includes ϕ and λ, while the state space $S = X \times [-\pi, \pi]$ considers heading angle as well. The problem is solved in a scenario-based environment obtained from realistic weather data using the Parallelized SB-RRTs.

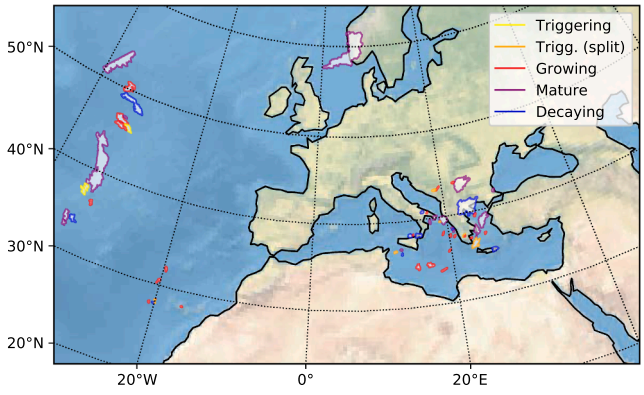
4.1. Problem setup

The test area consists of a stormy region detected by the Rapid Developing Thunderstorms (RDT) system[5] on November 16th, 2017 at 6:00 Zulu time. RDT is a product developed by Meteo-France for the detection, monitoring and forecast of convective cells, which uses imagery obtained by Meteosat Second Generation satellites. It is able to characterize convective systems around Europe every 15 min with a horizontal resolution of 3 km. RDT output includes a list of convective objects (Fig. 5(a)), as well as their speed, direction of motion, phase (e.g. growing, decaying.) and a deterministic extrapolation into the future. This data is post-processed (González-Arribas et al., 2019b), as illustrated in Fig. 5, incorporating uncertainties in the cell motion and obtaining the probability map $p(\mathbf{x})$ shown in Fig. 5(c). The function p represents the probability that $\mathbf{x} \in X$ is in a storm, where $\mathbf{x} = (\lambda, \phi)$.

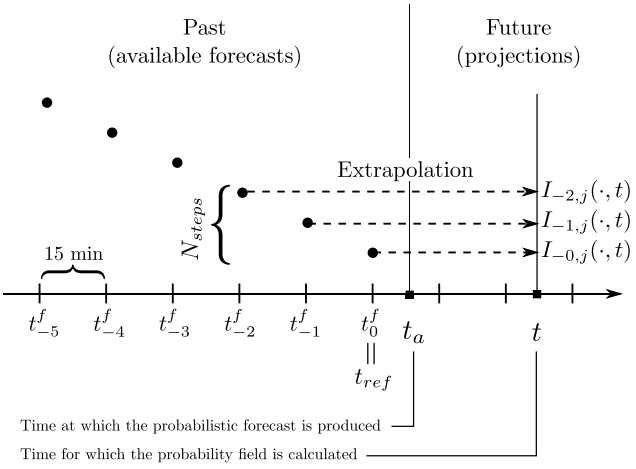
As stated in Bauer et al. (2015), to be able to capture storm cells more accurately, NWP is evolving towards convective-permitting ensemble prediction systems (EPS) of very high spatiotemporal resolution. These are not available yet but are expected in the near future. In consequence, to simulate an ensemble-based input and obtain different possible forecasts for the SB-RRTs, different ensemble members are sampled from the probability map in Fig. 6(a) as follows:

- The map is first discretized with a 0.1° step in latitude and longitude.
- For each position, a random number is taken from a uniform distribution between 0 and 1. If this number is lower than the actual probability of storm at the position, a storm is assigned to that position. For example, if the probability of having a storm is 90%, any sample between 0 and 0.9 corresponds to having a storm.
- The positions with a storm are clustered by means of a density-based clustering method, DBSCAN (Kröger et al., 2019).
- Finally, the polygon that encloses each cluster is calculated (Preparata and Hong, 1977).

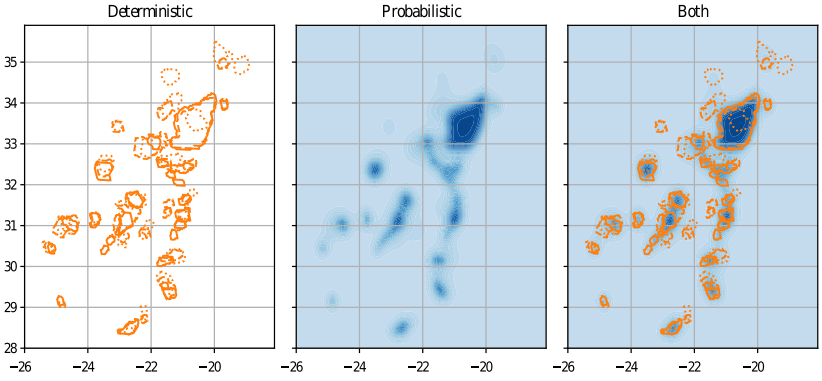
[5] http://www.nwcsaf.org/web/guest/nwc/geo-geostationary-near-real-time-v2018.



(a) Example of convective systems identified by RDT.



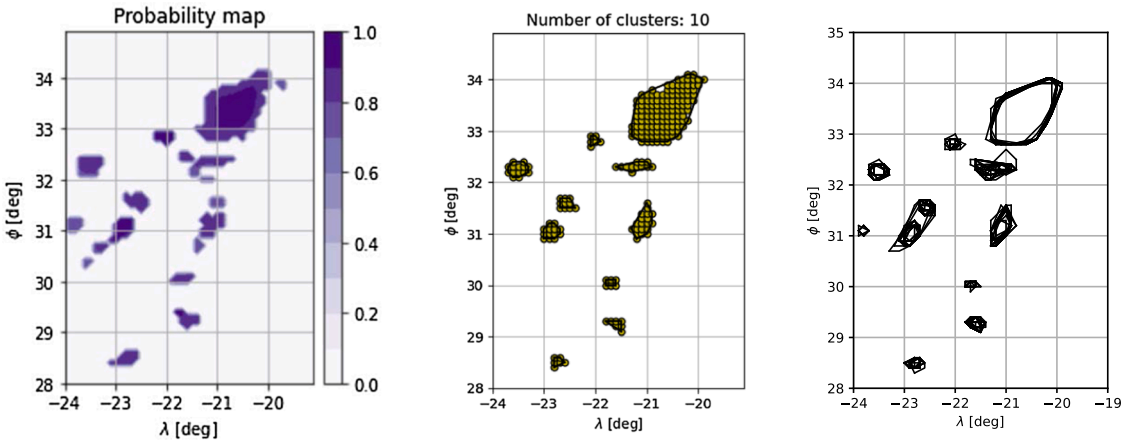
(b) Time-lagged ensemble methodology.



(c) Probabilistic storm model compared to the deterministic RDT data.

Fig. 5. Methodology to build a probabilistic storm model $p(\mathbf{x})$ based on RDT forecasts by means of a time-lagged ensemble forecast (see González-Arribas et al., 2019b).

An example of the process is shown in Fig. 6(b). As the result is a group of polygons, the intersection of each polygon with a RRT edge, if it exists, can be calculated by means of geometric operations. According to the formulation presented in SubSection 3.1, Fig. 6 (b) represents a set X^j_{unsafe}, with $j = 1,\ldots,N_{sc}$. Each O^j_l, with $l = 1,\ldots,10$, is a convex polygon. The process is then repeated N_{sc} times to obtain the desired number of scenarios. Fig. 6(c) shows an example of X_{unsafe} with 20 ensemble members.



(a) Probability map. (b) Example of sample. (c) Ensemble.

Fig. 6. Probability map (left), sample taken from it (center), and ensemble of 20 members (right).

In order to test the ability of the three SB-RRTs to avoid hazardous and uncertain regions, a safe route connecting $\mathbf{x}_{start} = (-22°, 34°)$ and $\mathbf{x}_{goal} = (-20°, 29°)$ is sought. The aircraft is assumed to maintain constant altitude at FL300. While the SB-RRT searches for safe trajectories, the SB-RRT* and the Informed SB-RRT* look for the optimal trajectory that minimizes flight distance.

4.2. Results

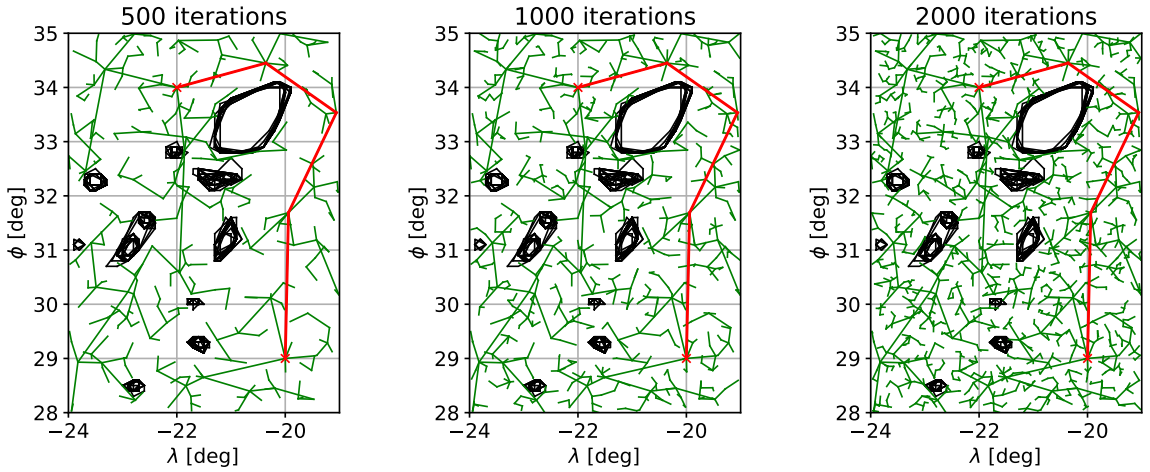
There are two main parameters that affect the performance of the three SB-RRTs:

- The maximum number of iterations *MaxIter*: as two of the algorithms are based on asymptotically optimal methodologies, a higher number of iterations leads to a better solution in terms of flight distance in exchange for higher computational times.
- The product $\lfloor \epsilon N_{sc} \rfloor$: this integer variable represents the maximum number of intersections allowed for each trajectory. A higher value means that the algorithm is able to grow closer to the unsafe regions. N_{sc} is given by the meteorological product while ϵ varies from 0 to 1.

Although it is not discussed here, there is a third parameter with influence on optimal algorithms, i.e. the constant μ that appears in *Near* function. The RRT* requirement for optimality is that $\mu > \mu^*$, with $\mu^* = \left(\frac{\log card(\mathcal{A})}{card(\mathcal{A})}\right)^{\frac{1}{d_x+1}}$. The value of μ is related to the number of nodes considered in the optimization process performed by *Parent* and *Rewire*. A higher value means that A_{near} covers a larger region and more nodes are likely to lie inside it. Consequently, the *for* loops in Algorithms 3 and 4 involve more edges to rearrange, and therefore higher computational times. In our simulations, a value of $\mu = 1.1\mu^*$ is chosen. A larger value degraded the computational performance with small benefits on the convergence process.

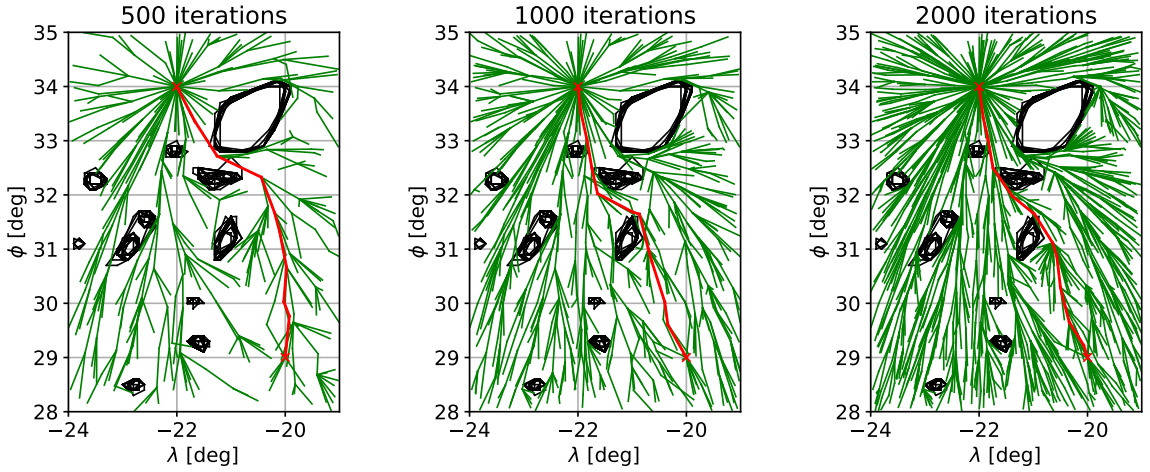
Number of iterations. To begin with, the influence of the maximum number of iterations is analyzed, assuming that $\epsilon = 0.1$ and $N_{sc} = 20$. First, the results corresponding to the SB-RRT after 500, 1000 and 2000 are shown in Fig. 7. As can be observed, there is no optimization, and once a path is found the algorithm will not further refine it irrespective of the number of iterations. This algorithm is much faster than the other two and would be useful in problems with no cost penalties in which only a safe trajectory is required. However, in aircraft trajectory planning problems, fuel, distance or time are objectives to be minimized and the optimal approaches must be employed. An example of SB-RRT* growth is included in Fig. 8. Note that the stopping criteria once a_{goal} is reached (Algorithm 1, line 12) is ignored with the purpose of showing that there is no optimization.

In the SB-RRT*, and in those algorithms based on the RRT*, there are two processes happening simultaneously, exploration and optimization, both of which can be observed in Fig. 8. On the one hand, the exploration results in reaching the highest possible number of nodes. Higher number of nodes implies that new samples are more likely to get connected to the tree. Exploration is associated to *RandomSample*, *NearestNode* and *Safe* (lines 4, 5 and 7, respectively, in Algorithm 2), which determine the new sample and the possibility of a connection towards it. On the other hand, the optimization seeks to lower the cost associated with the sequence of edges. Optimization is connected to *BestParent* and *Rewire* (lines 9 and 12, respectively, in Algorithm 2), both highly influenced by the number of nodes in A_{near}. The tree growth can be divided into two phases, depending on the predominance of exploration or optimization:



(a) 500 iterations. (b) 1000 iterations. (c) 2000 iterations.

Fig. 7. SB-RRT expansion (green) and solution (red) after 500, 1000 and 2000 iterations considering $\epsilon = 0.1$ and $N_{sc} = 20$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

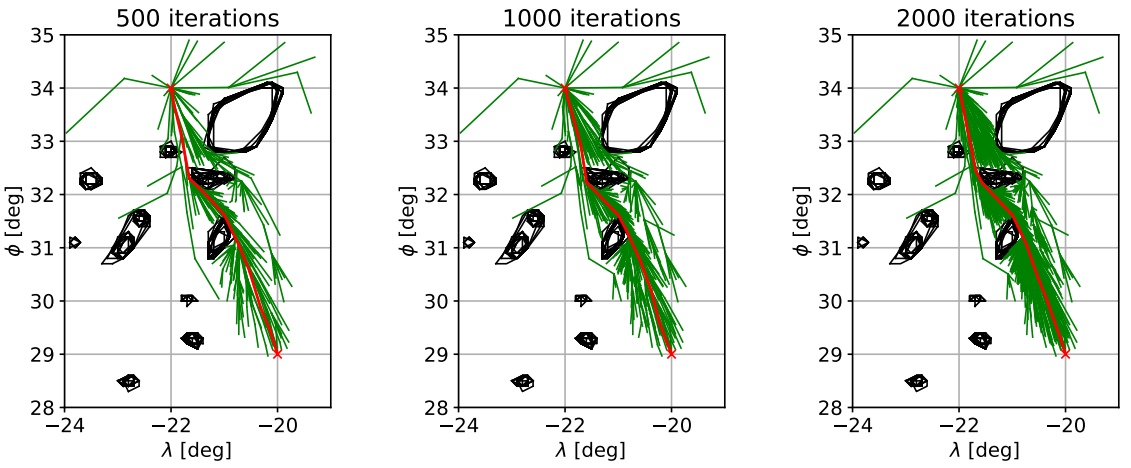


(a) 500 iterations. (b) 1000 iterations. (c) 2000 iterations.

Fig. 8. SB-RRT* expansion (green) and solution (red) after 500, 1000 and 2000 iterations considering $\epsilon = 0.1$ and $N_{sc} = 20$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

- In the early stages (Fig. 8(a)), exploration predominates in order to increase the density of nodes. During the initial iterations, if the algorithm is able to find a feasible trajectory, it is usually far from optimal. The exploration rate is small as the number of iterations is still insufficient to develop a proper tree. Many random samples are taken but not connected to the tree, due to the fact that no safe path to the existing branches is found and the sample tends to be rejected.
- In the later stages (Figs. 8(b)–(c)) the domain has been widely explored and the algorithm focuses on the optimization. As the number of iterations increases and more nodes are added to the tree, the edges are able to reach further areas and grow through the narrow corridors between unsafe regions. Consequently, this increase in the number of nodes facilitates the addition of new ones, more areas are explored and the probability of taking a random sample close to an existing branch increases. As can be seen, most of the nodes in the vicinity of $\mathbf{x}_{start}$ tend to be connected with a straight path, which reduces the distance as much as possible. With a large number of iterations the optimization becomes more effective, and this trend extends progressively towards more distant nodes.

As can be observed in Fig. 8, the SB-RRT* is optimizing all the paths that are connected to $\mathbf{x}_{start}$, a highly inefficient process. The Informed SB-RRT* shown in Fig. 9 addresses this problem. During the first iterations, the Informed SB-RRT* is identical to the SB-RRT*, growing to find a first path. Once it is found, the search region is reduced to an ellipsoid, as can be appreciated in Fig. 9(a)–(c). Inside the new search space, the aforementioned exploration and optimization processes take place as in the SB-RRT*.



(a) 500 iterations. (b) 1000 iterations. (c) 2000 iterations.

Fig. 9. Informed SB-RRT* expansion (green) and solution (red) after 500, 1000 and 2000 iterations considering $\epsilon = 0.1$ and $N_{sc} = 20$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Computational time (in seconds) as a function of the maximum number of iterations for $N_{sc} = 20$.

MaxIter	200	500	1000	2000	5000
SB-RRT	1.5	3.7	8.3	22.0	105.6
SB-RRT*	3.8	16.5	54.4	182.4	978.0
Inf. SB-RRT*	5.2	24.3	82.2	287.4	1530.0

A comparison of computational times with respect to the number of iterations is presented below. The average time at different iterations is included in Table 2.[6] A first conclusion is that computational time grows as $\sim Iterations^n$ with $n \sim 1.7 - 1.8$, hence doubling the iterations would require ~ 3.4 times more time. This holds for both the optimal algorithms and for the SB-RRT after 1000 iterations. A second observation is that, for more than 1000 iterations, the SB-RRT is an order of magnitude less time than the SB-RRT* and the Informed RRT*. This additional time is the cost of optimization. Note that running the SB-RRT for more than 100 iterations is not worthwhile. A first solution is usually found in less than 100 iterations, and increasing the number of iterations will not produce any improvement.

Safety margin and number of scenarios. As stated above, the influence of ϵ and N_{sc} is encapsulated in the product $\lfloor \epsilon N_{sc} \rfloor$. This variable measures the maximum number of intersections between any safe trajectory that starts in x_{start} and the set of scenarios in X_{unsafe}. Note that according to Eq. (14), each node in the tree verifies that $M_k \leqslant \lfloor \epsilon N_{sc} \rfloor$. Let the cost of safety be the relative difference between the distance covered by a solution and the great circle distance from the origin to the destination. The orthodrome would be the shortest path with no safety constraints. However, under the influence of thunderstorms, and in exchange for safety, the aircraft must deviate from this path, which leads to an increase in cost. Consequently, this product can also be considered as a measure of how conservative the simulation is. By tuning the value of $\lfloor \epsilon N_{sc} \rfloor$, the methodology allows a certain risk to reduce costs. The particular case of $\lfloor \epsilon N_{sc} \rfloor = 0$ would correspond to the most conservative scenario. Such problem is equivalent to a trajectory planning problem in a deterministic environment built as the Boolean union of all the scenarios.

Fig. 10 represents the average cost of safety for different values of $\lfloor \epsilon N_{sc} \rfloor$. Note that the SB-RRT is not included as it does not look for optimal solutions. Two trends can be observed. First, the convergence of both algorithms. As the number of iterations increase, the average cost of safety decreases. This effect is more abrupt in the Informed SB-RRT*, as convergence happens earlier and the curves for 500 and 1000 iterations almost overlap. Second, the asymptotic behavior with $\lfloor \epsilon N_{sc} \rfloor$. For larger values of this parameter, more intersections with X_{unsafe} are allowed, which permits the aircraft to get closer to the cells, thereby reducing total cost. However, for $\lfloor \epsilon N_{sc} \rfloor > 2$, it can be appreciated that this cost is almost constant. The main reason is that the different realizations that describe a storm cell are close together, so that intersecting more scenarios merely affects cost. For this purpose, in simulations with $N_{sc} = 20$, a safety margin $\epsilon = 0.1$ is chosen. Increasing it would not reduce the cost of safety significantly.

Fig. 11 represents the cost of safety for 20 different simulations and for $\lfloor \epsilon N_{sc} \rfloor$ between 0 and 10 after 1000 iterations. Again, it can be observed that the Informed SB-RRT* converges earlier than the SB-RRT*, as the cost presents less deviation from the mean value.

[6] Note that this is a proof of concept prototype in Python; the algorithm for operational use would be more efficient and computational times would be reduced.

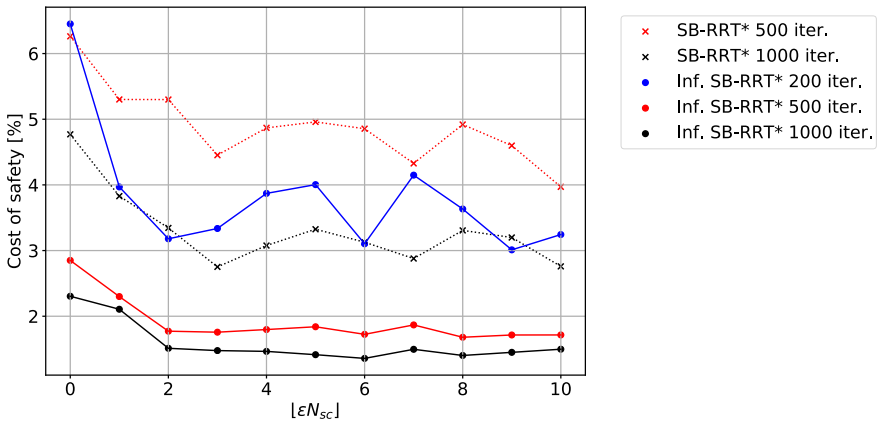


Fig. 10. Mean cost of safety as a function of the maximum number of intersections allowed $\lfloor \epsilon N_{sc} \rfloor$. SB-RRT* and Informed SB-RRT* are considered for 200, 500 and 1000 iterations.

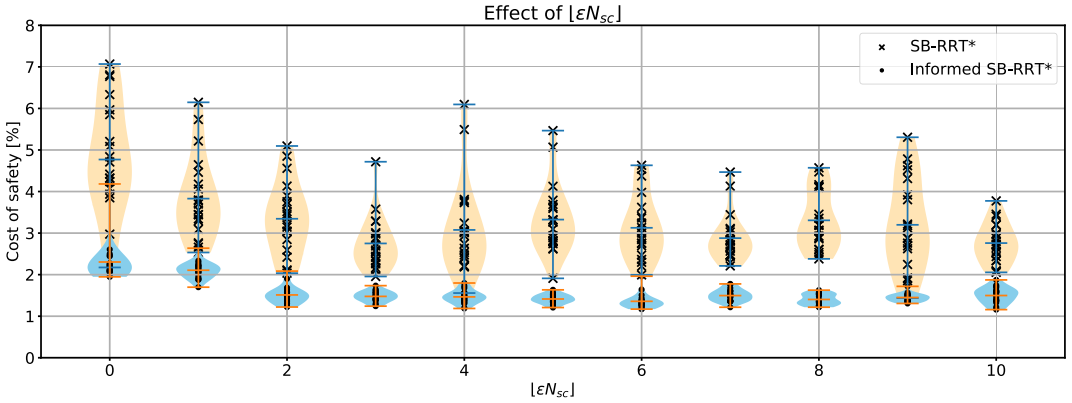


Fig. 11. Cost of safety for the SB-RRT* and the Informed SB-RRT* after 1000 iterations as a function of $\lfloor \epsilon N_{sc} \rfloor$.

4.3. Sensitivity of the results with respect to the number of iterations

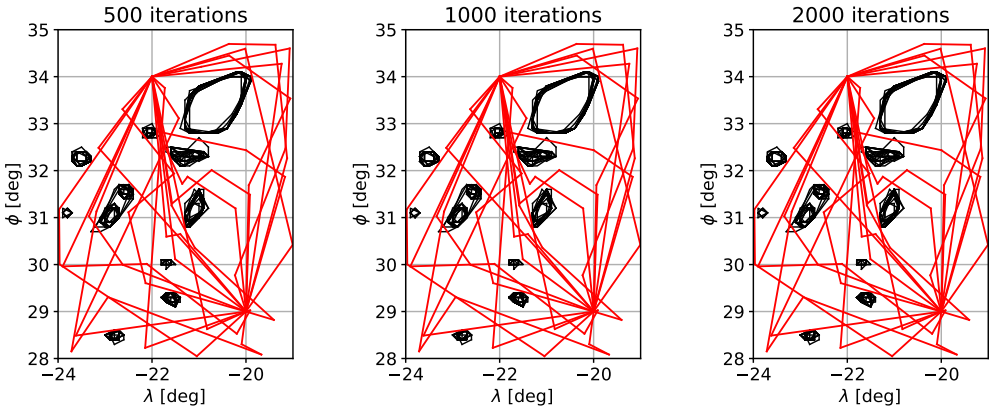
RRTs are meta-heuristic algorithms that explore a region based on a random sampling process. During each simulation the tree growth is different and, after the same number of iterations, solutions will differ between them. To show this variability, the different solutions provided by 20 runs of the SB-RRT with $\epsilon = 0.1$ and $N_{sc} = 20$ are included in Fig. 12. As can be observed, this algorithm is not influenced by the number of iterations; each simulation provides a safe trajectory and there are no further changes once it is found. The main advantage of this algorithm is its ability to find different trajectories in a reduced time (from Table 2, 1.5 s) but with no guarantees of optimality.

As stated earlier, the SB-RRT* and the Informed SB-RRT* are based on asymptotically optimal algorithms. Since it is impossible to run the code indefinitely, the sensitivity of the simulation with respect to the number of iterations is analyzed. To this end, the smallest value of *MaxIter* that presents a reduced variability in the results is sought. To better visualize this variability and the convergence of both algorithms, 20 routes obtained for *MaxIter* $= 500$, 1000 and 2000 with $\epsilon = 0.1$ and $N_{sc} = 20$ are shown in Figs. 13 and 14.

The analysis of the SB-RRT* results reveals that during the first series of iterations (Figs. 13(a)–(b)) there is a wide spectrum of safe solutions, but none of them are optimal. As the iterations increase (Figs. 13(c)), it is noticeable that the algorithm converges to 3 possible solutions, thus indicating the existence of different local optima. This is indeed a positive fact, as different alternatives, all of them safe, can be proposed to both pilots and air traffic controllers in times compatible with practical settings (seconds/minutes).

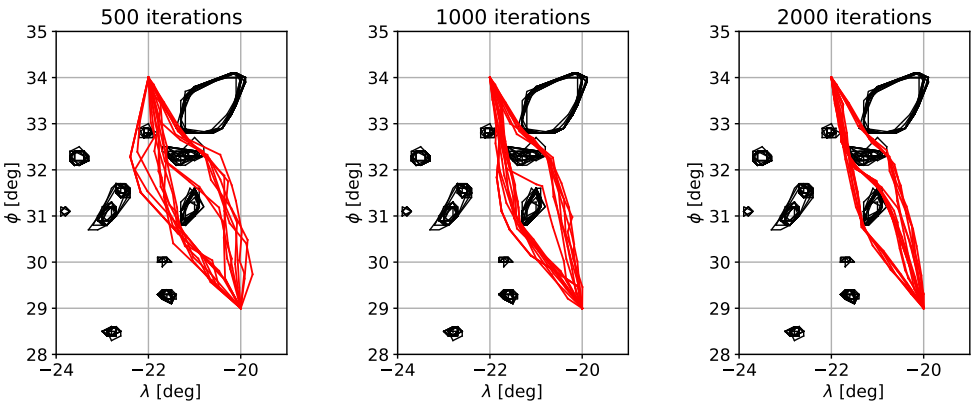
Contrariwise, the trajectories provided by the Informed SB-RRT* converged to the global optimum (in terms of flight distance) after a lower number of iterations (Figs. 14(a)–(c)). All the SB-RRT* simulations would eventually converge to this same solution but involving a larger number of iterations. This fact demonstrates the higher efficiency of the informed approach.

After a first qualitative analysis, the aforementioned cost of safety is represented with respect to the number of iterations for both algorithms. To illustrate this, a percentile representation of the results is shown in Fig. 15, including different color bands for the 0 and 100 percentiles and the median highlighted with a black line. The region of interest between 500 and 1000 iterations is zoomed on the right. It can be observed that as the maximum number of iterations is increased, the cost of safety decreases, and so does the variability in the solutions. These results allow the conclusion that it might not be necessary to run the algorithms for a large number of iterations.



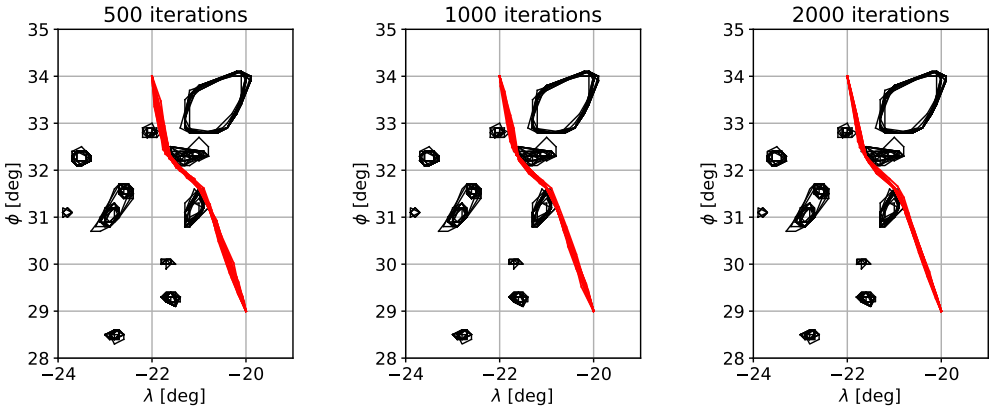
(a) 500 iterations. (b) 1000 iterations. (c) 2000 iterations.

Fig. 12. Sensitivity of the SB-RRT to the maximum number of iterations.



(a) 500 iterations. (b) 1000 iterations. (c) 2000 iterations.

Fig. 13. Sensitivity of the SB-RRT* to the maximum number of iterations.



(a) 500 iterations. (b) 1000 iterations. (c) 2000 iterations.

Fig. 14. Sensitivity of the Informed SB-RRT* to the maximum number of iterations.

On the one hand, the highest costs obtained by the SB-RRT* after 500 and 1000 iterations were 10.4% and 6.7%, respectively. In half of the simulations, the error is less than 4.7% and 3.1% after 500 and 1000 iterations, respectively. On the other hand, as was to be expected, the Informed SB-RRT* presented less variability for the same number of iterations. The highest costs were 2.4% and 2.0% after 500 and 1000 iterations, respectively. Additionally, the median is 1.7% after 500 iterations and 1.5% after 1000.

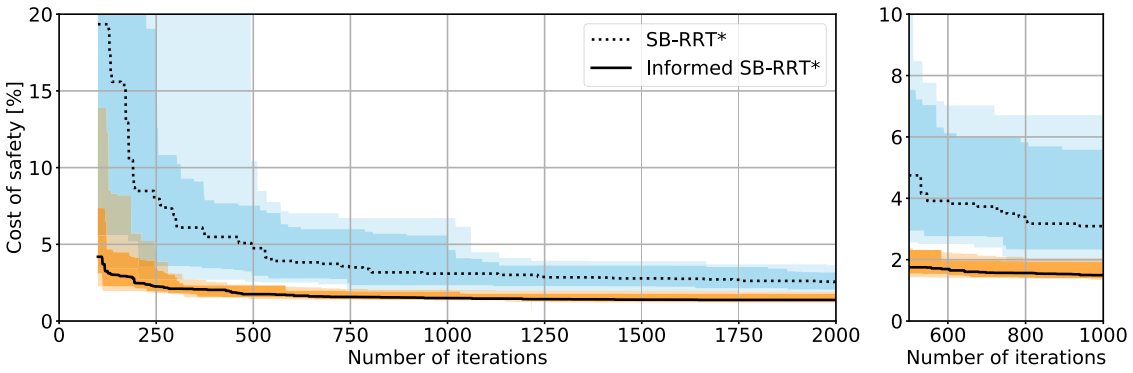
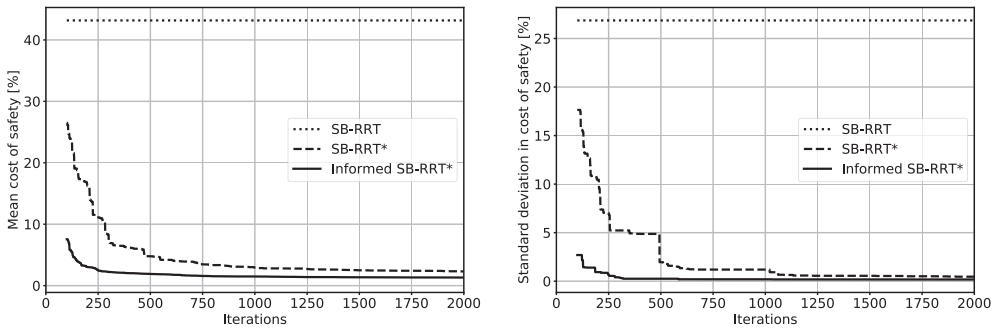


Fig. 15. Cost of safety as a function of the number of iterations for the SB-RRT* and the Informed SB-RRT*.



(a) Mean cost. (b) Standard deviation.

Fig. 16. Mean cost of safety and standard deviation with respect to the number of iterations.

To conclude, the mean cost of safety and its standard deviation are illustrated for all the SB-RRTs in Fig. 16. On one side, the SB-RRT presented constant and large values since there was no convergence and only provided random solutions. Conversely, the different convergence rates for the SB-RRT* and the Informed SB-RRT* can be appreciated. Fig. 16(b) shows that the informed algorithm almost eliminates the variability after 300 iterations, while the SB-RRT* required more than 1000. Although the latter is a slightly faster algorithm (see Table 2), it requires more iterations to reach the same convergence levels. By way of example, the 300 iterations of the Informed SB-RRT* take 10 s, whereas running the SB-RRT* for 1000 iterations takes around 1 min.

5. Conclusions

In this work, the scenario-based methodology for RRTs has been presented and applied to the RRT, the RRT* and the Informed RRT*. This results in the so-called Scenario-Based RRTs, three algorithms for aircraft flight planning in areas of uncertain convective weather. Each of them is able to find a safe trajectory that connects two states constrained by a safety margin ϵ. Additionally, the SB-RRT* and the Informed SB-RRT* are used to minimize flight distance and increase efficiency. In anticipation of future NWP products, an ensemble of possible weather forecasts is simulated to characterize the environmental uncertainties. For this particular example, the trees grow with a 90% probability of being safe. After 1000 iterations, the SB-RRT* provides solutions that increase $1.9-6.7\%$ with respect to the great circle distance, while the Informed SB-RRT* reduces this difference to $1.3-2.0\%$. The orthodrome, however, is unsafe with probability 1. Moreover, relying on parallel GPU programming, the algorithm is able to produce close-to-optimum solutions in seconds or minutes, being compatible with near real time operation.

The main disadvantage of the SB-RRT* is that it is based on an asymptotically optimal algorithm. Theoretically, a RRT* simulation would require an infinite number of iterations to reach the optimum. This issue is overcome with the informed algorithm that biases the search around the solution and only takes random samples that support the optimization. It is demonstrated that this focused sampling increases the efficiency substantially. As an example, 300 iterations are enough for the Informed SB-RRT* to converge, which takes around 10 s, while the SB-RRT* requires more than 1000 iterations (or 1 min). In the future, the effect of the number of storm cells and their arrangement on the convergence rate will be explored, as the number of iterations to approach close-to-optimum solutions is affected. Additionally, the formulation will be extended to consider not only weather but restricted areas or congested sectors.

Acknowledgment

This work has received funding from (1) the Spanish Government (Project RTI2018-098471-B-C32) and (2) the SESAR Joint Undertaking under the European Union's Horizon 2020 research and innovation programme under grant agreement No 783287. The opinions expressed herein reflect the authors' view only. Under no circumstances shall the SESAR Joint Undertaking be responsible for any use that may be made of the information contained herein. Funding for APC: Universidad Carlos III de Madrid (Read & Publish Agreement CRUE-CSIC 2021).

References

Andrés, E., González-Arribas, D., Sanjurjo-Rivo, M., Soler, M., Kamgarpour, M., 2020. GPU-accelerated RRT for flight planning considering ensemble forecasting of thunderstorms. In: SESAR Innovation Days 2020.

Aoude, G.S., Luders, B.D., Joseph, J.M., 2013. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion planning. Auton Robot 35, 51–76. https://doi.org/10.1007/s10514-013-9334-3.

Bauer, P., Thorpe, A., Brunet, G., 2015. The quiet revolution of numerical weather prediction. Nature 525, 47–55. https://doi.org/10.1038/nature14956.

Bhattacharya, S., Ghrist, R., Kumar, V., 2015. Persistent homology for path planning in uncertain environments. IEEE Trans. Rob. 31 (3), 578–590. https://doi.org/10.1109/TRO.2015.2412051.

Blum, C., Roli, A., 2001. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. 35, 268–308. https://doi.org/10.1145/937503.937505.

Bouttier, F., Vié, B., Nuissier, O., Raynaud, L., 2012. Impact of stochastic physics in a convection-permitting ensemble. Mon. Weather Rev. 140 (11), 3706–3721. https://doi.org/10.1175/MWR-D-12-00031.1. https://journals.ametsoc.org/view/journals/mwre/140/11/mwr-d-12-00031.1.xml.

Chen, J., Yousefi, A., Krishna, S., Sliney, B., Smith, P., 2012. Weather avoidance optimal routing for extended terminal airspace in support of dynamic airspace configuration. In: 2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC). https://doi.org/10.1109/DASC.2012.6382301.

Courchelle, V., Soler, M., González-Arribas, D., Delahaye, D., 2019. A simulated annealing approach to 3d strategic aircraft deconfliction based on en-route speed changes under wind and temperature uncertainties. Transp. Res. Part C: Emerg. Technol. 103, 194–210. https://doi.org/10.1016/j.trc.2019.03.024. http://www.sciencedirect.com/science/article/pii/S0968090X18305849.

Dadkhah, N., Mettler, B., 2012. Survey of motion planning literature in the presence of uncertainty: Considerations for UAV guidance. J. Intell. Rob. Syst. 65, 233–246. https://doi.org/10.1007/s10846-011-9642-9.

Duindam, V., Xu, J., Alterovitz, R., Sastry, S., Goldberg, K., 2010. Three-dimensional motion planning algorithms for steerable needles using inverse kinematics. Int. J. Rob. Res. 29, 789–800. https://doi.org/10.1177/0278364909352202.

Eurocontrol, 2020. Performance review report. An assessment of air traffic management in Europe during the calendar year 2019. Tech. rep.

Evans, A.D., Lee, P.U., 2019. Using machine-learning to dynamically generate operationally acceptable strategic reroute options. In: 13th USA/Europe Air Traffic Management Research and Development Seminar (ATM2019).

Forster, C., Ritter, A., Gemsa, S., Tafferner, A., Stich, D., 2016. Satellite-based real-time thunderstorm nowcasting for strategic flight planning en route. J. Air Transp. 24 (4), 113–124. https://doi.org/10.2514/1.D0055.

Frazzoli, E., Dahleh, M.A., Feron, E., 2002. Real-time motion planning for agile autonomous vehicles. AIAA J. Guid. Contr. Dyn. 25 (1), 116–129. https://doi.org/10.2514/2.4856.

Fulgenzi, C., 2009. Autonomous navigation in dynamic uncertain environment using probabilistic models of perception and collision risk prediction (PhD thesis). Institut National Polytechnique de Grenoble.

Gammell, J.D., Barfoot, T.D., Srinivasa, S.S., 2017. Informed sampling for asymptotically optimal path planning. IEEE Trans. Rob. 34. http://arxiv.org/abs/1706.06454.

Ghosh, D., Nandakumar, G., Narayanan, K., Honkote, V., Sharma, S., 2019. Kinematic constraints based bi-directional RRT (KB-RRT) with parameterized trajectories for robot path planning in cluttered environment. In: International Conference on Robotics and Automation, ICRA 2019. https://doi.org/10.1109/ICRA.2019.8793896.

Gong, C., McNally, D., 2015. Dynamic arrival routes: A trajectory-based weather avoidance system for merging arrivals and metering. In: 15th AIAA Aviation Technology, Integration and Operations Conference. doi:10.2514/6.2015-3394.

González-Arribas, D., Sanjurjo-Rivo, M., Soler, M., 2019. Multiobjective Optimisation of Aircraft Trajectories Under Wind Uncertainty Using GPU Parallelism and Genetic Algorithms. Springer International Publishing, pp. 453–466. doi:10.1007/978-3-319-89890-2_29.

González-Arribas, D., Soler, M., Sanjurjo-Rivo, M., Kamgarpour, M., Simarro, J., 2019b. Robust aircraft trajectory planning under uncertain convective environments with optimal control and rapidly developing thunderstorms. Aerosp. Sci. Technol. 89, 445–459. https://doi.org/10.1016/j.ast.2019.03.051.

Henderson, J., 2013. The Traffic Aware Strategic Aircrew Requests (TASAR) Concept of Operations. Tech. rep.. Engility Corporation for the National Aeronautics and Space Administration.

Hentzen, D., Kamgarpour, M., Soler, M., González-Arribas, D., 2018. On maximizing safety in stochastic aircraft trajectory planning with uncertain thunderstorm development. Aersop. Sci. Technol. 79, 543–553. https://doi.org/10.1016/j.ast.2018.06.006.

Hong, Y., Choi, B., Kim, Y., 2019. Two-stage stochastic programming based on particle swarm optimization for aircraft sequencing and scheduling. IEEE Trans. Intell. Transp. Syst. 20 (4), 1365–1377.

Isaacson, D., Gong, C., 2018. Dynamic Routes for Arrivals in Weather (DRAW) Concept of Operations, Tech. rep. National Aeronautics and Space Administration.

Karaman, S., Frazzoli, E., 2011. Sampling-based algorithms for optimal motion planning. Int. J. Rob. Res. 30, 846–894. https://doi.org/10.1177/0278364911406761.

Kessinger, C, Blackburn, G., Rehak, N., Ritter, A., Milczewski, K., Sievers, K., Wolf, D., 2015. Demonstration of a convective weather product into the flight deck. In: American Meteorological Society, 2015. URL https://ams.confex.com/ams/95Annual/webprogram/Paper269015.html.

Kröger, P., Jörg, S., Zimek, A.2019. Density-based clustering, WIREs Data Mining Knowl Discov. doi: 10.1002/widm.30.

Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., How, J.P., 2009. Real-time motion planning with applications to autonomous urban driving. IEEE Trans. Control Syst. Technol. 17 (5), 1105–1118. https://doi.org/10.1109/TCST.2008.2012116.

LaValle, S.M., 1998. Rapidly-exploring random trees: A new tool for path planning. Tech. rep.. Iowa State University.

LaValle, S.M., Kuffner, J.J., 2001. Randomized kinodynamic planning. In: The International Journal of Robotics Research, vol. 20, pp. 378–400. doi: 10.1177/02783640122067453.

Lefkopoulos, V., Kamgarpour, M., 2019. Using uncertainty data in chance-constrained trajectory planning. In: 2019 18th European Control Conference (ECC). https://doi.org/10.23919/ecc.2019.8795823.

Lunnon, R.W., Hauf, T., Gerz, T., Josse, P., 2009. FLYSAFE meteorological hazard nowcasting driven by the needs of the pilot. In: American Meteorological Society. URL https:/ams.confex.com/ams/pdfpapers/103462.pdf.

Matthews, M., Delaura, R., 2010. Assessment and interpretation of en route weather avoidance fields from the convective weather avoidance model. doi: 10.2514/6.2010-9160.

McNally, D., Sheth, K., Gong, C., Sterenchuk, M., Sahlman, S., Hinton, S., Lee, C.H., Shih, F.T., 2015. Dynamic weather routes: Two years of operational testing at american airlines. Air Traff. Contr. Quart. 23 (1), 55–81. https://doi.org/10.2514/atcq.23.1.55.

Moon, C., Chung, W., 2015. Kinodynamic planner dual-tree RRT (DT-RRT) for two-wheeled mobile robots using the rapidly exploring random tree. IEEE Trans. Industr. Electron. 62 (2), 1080–1090. https://doi.org/10.1109/TIE.2014.2345351.

Ng, H., Grabbe, S., Mukherjee, A., 2009. Design and evaluation of a dynamic programming flight routing algorithm using the convective weather avoidance model. doi: 10.2514/6.2009-5862.

NVIDIA Corporation, 2010. CUDA Programming Guide.

WMO, 2012. Guidelines on Ensemble Prediction Systems and Forecasting, WMO (Series). World Meteorological Organization.

Pharpatara, P., Hérissé, B., Bestaoui, Y., 2017. 3-D trajectory planning of aerial vehicles using RRT*. IEEE Trans. Control Syst. Technol. 25, 1116–1123. https://doi.org/10.1109/TCST.2016.2582144.

Preparata, F.P., Hong, S.J.. 1977. Convex hulls of finite sets of points in two and three dimensions. Commun. ACM.

Seenivasan, D.B., Olivares, A., Staffetti, E., 2020. Multi-aircraft optimal 4D online trajectory planning in the presence of a multi-cell storm in development. Transp. Res. Part C: Emerg. Technol. 110, 123–142. https://doi.org/10.1016/j.trc.2019.11.014. http://www.sciencedirect.com/science/article/pii/S0968090X19302803.

Sheth, K.S., Madson, M., Harrison, S.J., Helton, D., 2018. Air Traffic Management Technology Demonstration - 3 (ATD-3). Operational Concept for the Integration of ATD-3 Capabilities, Tech. rep. National Aeronautics and Space Administration.

Soler, M., González-Arribas, D., Sanjurjo-Rivo, M., García-Heras, J., Sacher, D., Gelhardt, U., Lang, J., Hauf, T., Simarro, J., 2020. Influence of atmospheric uncertainty, convective indicators, and cost-index on the leveled aircraft trajectory optimization problem. Transp. Res. Part C: Emerg. Technol. 120, 102784. https://doi.org/10.1016/j.trc.2020.102784. http://www.sciencedirect.com/science/article/pii/S0968090X2030694X.

Solovey, K., Janson, L., Schmerling, E., Frazzoli, E., Pavone, M., 2020. Revisiting the asymptotic optimality of RRT*. https://arxiv.org/abs/1909.09688v2.

Taylor, C., Wanke, C., 2012. Dynamically generating operationally acceptable route alternatives using simulated annealing. Air Traff. Contr. Quart. 20 (1), 97–121. https://doi.org/10.2514/atcq.20.1.97.

Taylor, C., Liu, S., Wanke, C., Stewart, T., 2018. Generating diverse reroutes for tactical constraint avoidance. J. Air Transp. 26 (2), 49–59. https://doi.org/10.2514/1.D0089.