Solving Time Dependent Shortest Path Problems on Airway Networks Using Super-Optimal Wind *

Marco Blanco[1], Ralf Borndörfer[2], Nam-Dũng Hoang[3], Anton Kaier[4], Adam Schienle[5], Thomas Schlechte[6], and Swen Schlobach[7]

1 **Zuse Institute Berlin, Berlin, Germany**
`blanco@zib.de`
2 **Zuse Institute Berlin, Berlin, Germany**
`borndoerfer@zib.de`
3 **Zuse Institute Berlin, Berlin, Germany and Faculty of Mathematics, Mechanics and Informatics, Vietnam National University, Hanoi, Vietnam**
`hoang@zib.de`
4 **Lufthansa Systems GmbH & Co. KG, Kelsterbach, Germany**
`kaier@lhsystems.com`
5 **Zuse Institute Berlin, Berlin, Germany**
`schienle@zib.de`
6 **Zuse Institute Berlin, Berlin, Germany**
`schlechte@zib.de`
7 **Lufthansa Systems GmbH & Co. KG, Kelsterbach, Germany**
`schlobach@lhsystems.com`

Abstract

We study the Flight Planning Problem for a single aircraft, which deals with finding a path of minimal travel time in an airway network. Flight time along arcs is affected by wind speed and direction, which are functions of time. We consider three variants of the problem, which can be modeled as, respectively, a classical shortest path problem in a metric space, a time-dependent shortest path problem with piecewise linear travel time functions, and a time-dependent shortest path problem with piecewise differentiable travel time functions.

The shortest path problem and its time-dependent variant have been extensively studied, in particular, for road networks. Airway networks, however, have different characteristics: the average node degree is higher and shortest paths usually have only few arcs.

We propose A* algorithms for each of the problem variants. In particular, for the third problem, we introduce an application-specific "super-optimal wind" potential function that overestimates optimal wind conditions on each arc, and establish a linear error bound. We compare the performance of our methods with the standard Dijkstra algorithm and the Contraction Hierarchies (CHs) algorithm. Our computational results on real world instances show that CHs do not perform as well as on road networks. On the other hand, A* guided by our potentials yields very good results. In particular, for the case of piecewise linear travel time functions, we achieve query times about 15 times shorter than CHs.

1998 ACM Subject Classification G.1.0 General Numerical Analysis, G.1.6 Optimization, G.2.2 Graph Theory

* This work was partially supported by the BMBF program "Mathematics for Innovations in Industry and Services", under the project "E-Motion" (grant 05M12ZAB).

16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'16).
Editors: Marc Goerigk and Renato Werneck; Article No. 12; pp. 12:1–12:15

Open Access Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Keywords and phrases shortest path problem, A*, flight trajectory optimization, preprocessing, contraction hierarchies, time-dependent shortest path problem

Digital Object Identifier 10.4230/OASIcs.ATMOS.2016.12

1 Introduction

We consider the *Flight Planning Problem* (FPP), which seeks to compute a cost-minimal flight trajectory on an airway network, given origin and destination airports, a departure time, an aircraft, and weather prognoses. Some of the factors that need to be taken into account are overflight costs, weight-dependent fuel consumption functions, avoidance of hazardous areas, and restrictions to prevent overcrowding of airspaces, such as those published by EUROCONTROL in the *Route Availability Document* [9]. A comprehensive discussion of the FPP can be found in the survey [13]. However, to the best of our knowledge, the algorithmic treatment of flight planning problems on the complete airway network has not yet been considered in the literature. Existing approaches to the FPP, such as [5] or [6], consider only small regions of the airway network or artificial networks.

In this paper, we will focus on the *Horizontal Flight Planning Problem* (HFPP), a variant that seeks to minimize total flight time (in this case equivalent to total fuel consumption) while flying at constant altitude. This variant is very important because it is often used in practice as a subroutine in sequential approaches for computing 4-dimensional routes (with speed as the fourth dimension) [13]. Furthermore, it can be argued that the cruise phase is more important in terms of potential savings than the climb and descent phases, in particular for long-haul flights. Flight time between any two points is highly dependent on weather conditions, which are given as a function of time. For this reason, we model the HFPP as a Time-Dependent Shortest Path Problem (TDSPP).

The classical Shortest Path Problem (SPP) and the TDSPP have been extensively studied in the literature, with particular emphasis on routing in road networks. The past decades have seen a significant development of preprocessing techniques for both the SPP and the TDSPP, which yield astounding speedups compared to Dijkstra's algorithm, see [2], [8] for comprehensive surveys. Some of the most prominent state-of-the-art approaches are the following:

- The A* algorithm was first introduced in [12]. It is based on finding a *potential* function that, for each node, underestimates the length of an optimal path which uses it. The main ingredient for designing a potential function is thus an underestimator of the distance from each node to the target. In road networks, an obvious choice for such an underestimator is the *great circle distance* (GCD) to the target node. However, this method usually provides very loose underestimators (and thus very small speedups), due to the fact that subpaths of the optimal route often deviate substantially from the great circle connecting their endpoints. This can be explained by the grid-like topology of most road networks and the existence of obstacles such as rivers or mountain ranges. Therefore, more sophisticated potential functions have been developed, such as ALT, see the next item.
- A* *with Landmarks and Triangle-inequality* (ALT) [11] is a variant of the A* algorithm, which can also be extended to the time-dependent case [14]. The main idea is to identify a set of "important" nodes, known as *landmarks*, for which a one-to-all (or all-to-one) shortest path tree is computed. The potential of each node is then computed by using these stored distances and the triangle inequality. The main challenge is defining the landmarks, which should ideally lie on a large number of shortest paths, or close to them.

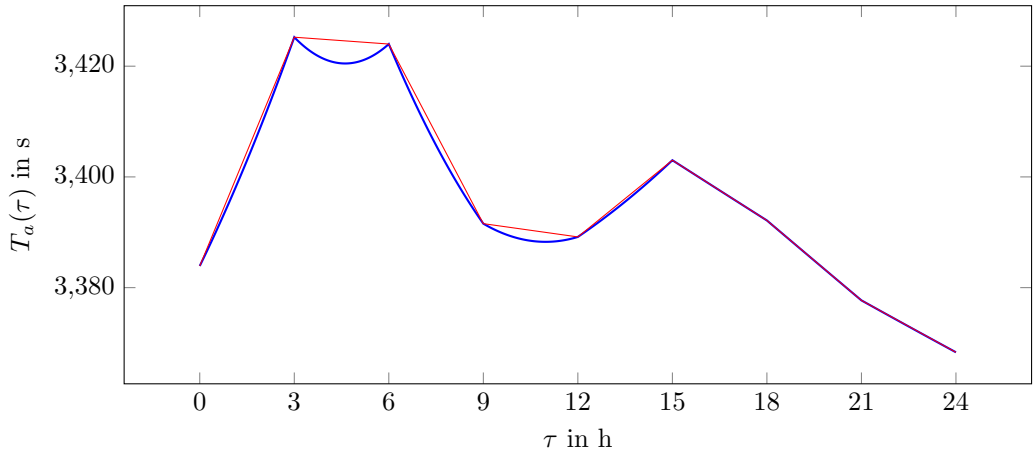


Figure 1 The exact travel time function $T(a, \tau)$ in red and the approximated piecewise linear function in blue, for some arc a.

The ALT algorithm can be further improved by combining it with other preprocessing techniques, see [4].

- *Contraction Hierarchies* (CHs) [10], as well as its time-dependent counterpart, Time-dependent Contraction Hierarchies (TCHs) [3], is one of the leading techniques in shortest paths computation. Even though TCHs have the disadvantage of large space requirements, they are considered one of the (if not *the*) best algorithms for the TDSPP in road networks [8], due to their lower preprocessing times. To the best of our knowledge, computational results on the performance of CHs and TCHs have been published only for road networks and public transportation networks [2].
- Approaches based on *Hierarchical Hub Labeling* [1] have been shown to be effective not only on road networks but on a large variety of input graphs [7], such as social networks or computer game networks. However, the nature of this approach seems to make it unsuitable for extension to the time-dependent case.

We will consider the real-world airway network. Its characteristics are very different from those of road networks. As of 2016, the complete horizontal network has approximately 53000 nodes and 330000 arcs after some preprocessing (i.e., contracting a large set of nodes with in-degree and out-degree equal to one). The average node degree of over six is higher than in road networks (usually between two and three), but still significantly smaller than in typical social networks (often in the two-digit range, see [7]). An advantage of flight planning over routing in road networks is that the number of possible OD-pairs is small. In fact, only about 1300 airports worldwide are used by commercial airlines[1]. Also, flight paths are typically short, usually involving below one hundred nodes, and do not deviate much from the great circle connection. It turns out that shortest path computation in airway networks is heavily influenced by these characteristics, and that the relative performance of the algorithms is different than in road networks.

In this paper, we investigate three variants of the HFPP.

- The *static* case is a particular shortest path problem, where the nodes belong to a metric space and arc costs are given by the corresponding distance (i.e., the GCD in our case).

[1] According to data from www.flightradar24.com

We will denote this problem as SPP. We present an A* algorithm in which the lower bounds for the potential function are given by the GCD from any node to the target node. This makes it possible to avoid the preprocessing step completely. Our computational results show that the speedup is comparable to that of CHs.

- The *exact dynamic* case is a Time Dependent Shortest Path Problem. In contrast to the literature standard, the time-dependent travel time functions (TTFs) on the arcs are not piecewise linear. In fact, in our application, the TTFs depend on wind forecasts and model the exact arc travel time. We refer to this problem as TDSPP-E. We present a *super-optimal wind* algorithm that underestimates the minimal travel time on each arc using the Newton method and establish a strong a priori error bound. The super-optimal wind bounds and the fact that the set of targets is known in advance, allow us to design an A* algorithm that yields a speedup of approximately 20 w.r.t. Dijkstra. Due to the non-linearity of the TTFs, this problem can not be solved by state-of-the-art TDSPP algorithms, in particular TCHs.
- Finally, in the *approximate dynamic* case, we consider a standard Time Dependent Shortest Path Problem. To this purpose, we approximate all TTFs by piecewise linear functions. Figure 1 shows an exact TTF and its approximate counterpart. We denote the resulting problem as TDSPP-PWL and present an A* algorithm similar to the one for TDSPP-E. Our computations show that the average speedup is approximately 25 with respect to Dijkstra, and 15 with respect to TCHs.

In Section 2, we describe the problems that we will study. In particular, we give a detailed description of the TTFs used in the exact dynamic case, to model the time-dependent influence of the wind on the travel time. Section 3 presents the super-optimal wind algorithm and the corresponding potential functions for the A* algorithm in the exact dynamic case. Finally, Section 4 presents computational results computed on real world data.

2 The Horizontal Flight Planning Problem

The HFPP can be modeled in terms of the Time-Dependent Shortest Path Problem, which is defined as follows: Given are a directed graph $D = (V, A)$ (In our application, nodes represent *waypoints* in the airway network and arcs stand for *airway segments*) and, for each $a \in A$, a *travel time function* (TTF) $T(a, \cdot) : [0, \infty) \to [0, \infty)$ that depends on the entering time. The travel time along a path $(v_0, v_1, \ldots, v_k)$ departing at time τ is defined as

$$T((v_0, \ldots, v_k), \tau) = \begin{cases} T((v_0, v_1), \tau) & k = 1 \\ T((v_0, \ldots, v_{k-1}), \tau) + T((v_{k-1}, v_k), T((v_0, \ldots, v_{k-1}), \tau) + \tau) & k > 1. \end{cases}$$

Given a pair of nodes $s, t \in V$ and a departure time $\tau \geq 0$, the objective is to find an s,t-path P in D such that the total travel time $T(P, \tau)$ is minimized.

The literature on the TDSPP usually considers piecewise linear (PWL) TTFs. We will denote this special (*approximate*) case of the dynamic problem as TDSPP-PWL.

The *exact* version of the dynamic problem, which we denote as TDSPP-E, assumes functions $T(a, \cdot)$ as described subsequently in Subsection 2.1. Finally, when $T(a, \cdot)$ is constant for every $a \in A$, we obtain the classical shortest path problem, denoted simply as SPP.

A standard assumption on TTFs is that they satisfy the *First-In-First-Out* (FIFO) property, which states that overtaking on arcs is not possible. That is, $T(a, \tau^1) \leq (\tau^2 - \tau^1) + T(a, \tau^2)$ for every $a \in A$, $0 \leq \tau^1 \leq \tau^2$. It is well known that the FIFO property guarantees correctness of the Dijkstra and A* algorithms, while the TDSPP is NP-hard in the general case. In the remainder of this paper, we assume that the FIFO property is always satisfied.

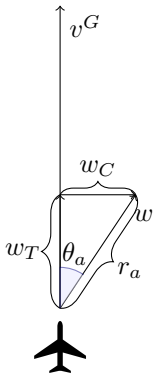


Figure 2 Crosswind and tailwind components.

2.1 Wind-Dependent Travel Time Functions

In this subsection, we define the travel time functions $T(a, \cdot)$ for the exact dynamic HFPP. We first recall some aeronautics terminology.

Let $a = (u, v) \in A$ and $\tau \geq 0$. For the next definitions, assume that the aircraft is located at u at time τ and then proceeds to traverse a. The *ground distance* $d^G(a)$ is defined as the GCD between u and v, and is thus independent of τ. The *airspeed* v^A is the speed relative to the air mass surrounding the aircraft. In our application, we assume that v^A is constant. Finally, the *ground speed* $v^G(a, \tau)$ is the aircraft's speed relative to the ground at the moment in which the aircraft enters arc a. The ground speed can be described in terms of a wind vector w acting on a at time τ as follows:

$$v^G(a, \tau) = \sqrt{(v^A)^2 - w_C(a, \tau)^2} + w_T(a, \tau).$$

Here, $w_C(a, \tau)$ represents the *crosswind component* and $w_T(a, \tau)$ the *tailwind component* affecting arc a at time τ; these are the components of the wind vector with angles $\frac{\pi}{2}$ and 0 with respect to a's direction, respectively, see Figure 2. Since an aircraft's airspeed is always much larger than wind speed, we can assume that v^G is always well-defined and positive.

Consider a wind vector $w(a, \tau) = \big(r_a(\tau), \theta_a(\tau)\big)$ acting on a at time τ, where $r_a(\tau)$ is the *wind speed*, i.e., the wind vector's magnitude; and $\theta_a(\tau)$ is the *wind direction*, i.e., the angle with respect to the arc's direction. Then, the crosswind and tailwind components can be computed as follows:

$$w_C(a, \tau) = r_a(\tau) \sin(\theta_a(\tau)) \quad \text{and} \quad w_T(a, \tau) = r_a(\tau) \cos(\theta_a(\tau)).$$

A weather prognosis set provides wind information for a finite number of time points $t_0 < t_1 < \cdots < t_N$. Without loss of generality, we will assume $t_0 = 0$. Furthermore, in practice, prognosis sets are used to plan flights taking off after time t_0 and landing well before time t_N. For that reason, in the rest of the paper, we will assume that we are only interested in evaluating TTFs for $\tau \in [t_0, t_N]$.

If $t_i < \tau < t_{i+1}$ for $i \in \{0, \ldots, N-1\}$, the wind vector is interpolated. More precisely, given two wind vectors w_i and w_{i+1} for arc a at times t_i and t_{i+1}, defined by wind speeds r_a^i, r_a^{i+1} and directions $\theta_a^i, \theta_a^{i+1}$, then for $\tau = \lambda t_i + (i - \lambda) t_{i+1}$ with $\lambda \in (0, 1)$, the wind vector at time τ is defined by

$$r_a(\tau) = \lambda r_a^i + (1 - \lambda) r_a^{i+1} \quad \text{and} \quad \theta_a(\tau) = \lambda \theta_a^i + (1 - \lambda) \theta_a^{i+1}.$$

Therefore the resulting crosswind and tailwind components at time τ are

$$w_C(a,\tau) = \left(\lambda r_a^i + (1-\lambda) r_a^{i+1}\right) \sin\left(\lambda\theta_a^i + (1-\lambda)\theta_a^{i+1}\right)$$
$$w_T(a,\tau) = \left(\lambda r_a^i + (1-\lambda) r_a^{i+1}\right) \cos\left(\lambda\theta_a^i + (1-\lambda)\theta_a^{i+1}\right).$$

In this paper, we assume that $w_C(a,\tau)$ and $w_T(a,\tau)$ remain constant during the traversal of a and define the travel time $T(a,\tau)$ across arc a entering at time τ as

$$T(a,\tau) = \frac{d^G(a)}{v^G(a,\tau)} = \frac{d^G(a)}{\sqrt{(v^A)^2 - w_C(a,\tau)^2} + w_T(a,\tau)}. \tag{1}$$

One can argue that the functions $T(a,\cdot)$ in (1) satisfy the FIFO property for realistic weather conditions. They represent the industrial state-of-the-art in aeronautical computations.

3 A* algorithms for the HFPP

For each of the three problem variants described in Section 2, we design an A* algorithm. Such an algorithm is based on a *potential* function $\pi : V \to \mathbb{R}$ that, for every $v \in V$, underestimates the cost of the shortest (v,t)-path. The potential is used to define the *reduced cost* of an arc (u,v) at time τ as follows:

$$T'((u,v),\tau) := T((u,v),\tau) - \pi(u) + \pi(v).$$

If $T'((u,v),\tau) \geq 0$ for every $(u,v) \in A$, $\tau \geq 0$, we say that π is *feasible*. Given this condition, the A* algorithm is equivalent to running Dijkstra on D using the reduced costs T'. In the following, we will introduce potential functions for each of the three problem variants.

3.1 Potential in the Static Case

In the static case, i.e., for SPP, a potential function for A* can be computed by simply considering the great-circle-distance between any node and the target node. That is, given $v \in V$ and a target node $t \in V$, we define

$$\pi(v) := d^G(v,t),$$

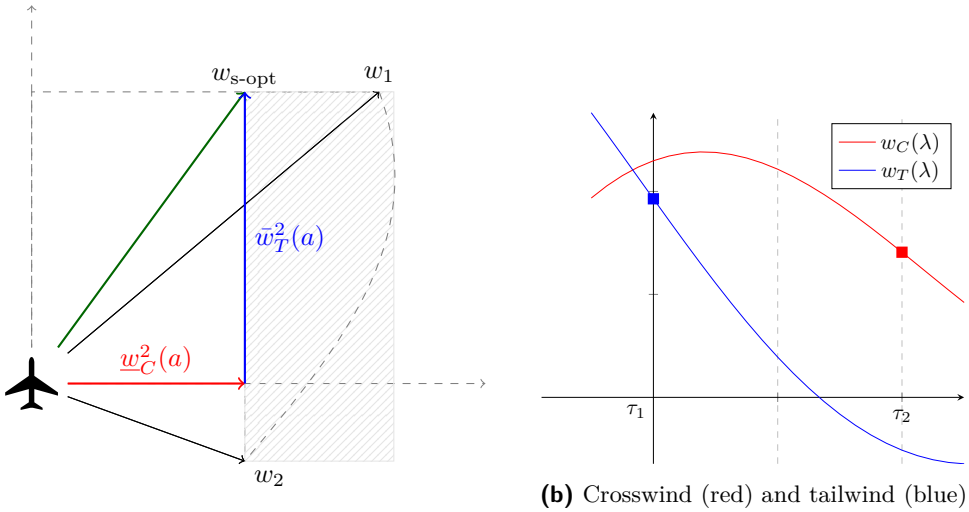
where $d^G : V \times V \to \mathbb{R}_+$ is the GCD-function. The advantage of this approach is that π can be computed on-the-fly during the query, and so no preprocessing step is necessary.

3.2 Potential in the Approximate Dynamic Case

For TDSPP-PWL, we make use of the fact that, in our application, there exists a small number of possible targets (which correspond to airports). Thus, we compute a lower bound on the minimum travel time from each node to each airport. For this, we first seek a value $\underline{T}(a)$ that, for each arc a, lower-bounds all possible travel times on the arc. That is, $\underline{T}(a) \leq T(a,\tau)$ for each $\tau \in [t_0, t_N]$. Since $T(a,\cdot)$ is a piecewise linear function, this bound can be found in linear time. Then, we compute all-to-one shortest path trees towards all airport nodes using $\underline{T}$ as arc costs and set

$$\pi^t(v) = \min\left\{\sum_{a\in P} \underline{T}(a) | P \text{ is a } (v,t)\text{-path}\right\} \tag{2}$$

for every node v and every possible target node t. Given an OD-pair s,t, we choose $\pi^t(\cdot)$ as a potential function. We remark that this is equivalent to choosing all airport nodes as landmarks in the ALT algorithm.



(a) The super-optimal wind vector $w_{\text{s-opt}}$ resulting from w_1 and w_2.

(b) Crosswind (red) and tailwind (blue) functions for the wind vectors in the interval $[\tau_1, \tau_2]$.

Figure 3 Super-optimal wind and component functions for wind vectors w_1 and w_2, corresponding to time points τ_1 and τ_2.

3.3 Potential in the Exact Dynamic Case Using Super-Optimal Wind

For TDSPP-E, we also compute lower bounds $\underline{T}$ on the TTFs and then define π according to (2). As opposed to the approximate case, finding good lower bounds $\underline{T}(a)$ is not straightforward. This section is dedicated to the solution of this problem.

It is clear from (1) that an upper bound on the ground speed directly leads to a lower bound on the travel time. Thus, to find a good lower bound on $T^*(a) := \min_{\tau\in[t_0,t_N]} T(a,\tau)$, we will concentrate on finding a good upper bound on $v_*^G(a) := \max_{\tau\in[t_0,t_N]} v^G(a,\tau)$.

We assume that the length of the weather prognosis intervals is constant, i.e., $t_i - t_{i-1} \equiv L > 0$ for $i = 1,\dots,N$. Our first step is to discretize the time interval $[t_0, t_N]$ into smaller intervals of length $\Delta > 0$. That is, we define $\tau_0,\dots,\tau_K$ such that $t_0 = \tau_0 < \tau_1 < \cdots < \tau_K = t_N$, $\Delta = \tau_k - \tau_{k-1}$ for $k = 1,2,\dots,K$; and N divides K. This condition guarantees that every two consecutive time points τ_{k-1} and τ_k belong to an interval $[t_{i-1}, t_i]$ for some index i. Define

$$\underline{w}_C^k(a) := \min_{\tau\in[\tau_{k-1},\tau_k]} |w_C(a,\tau)|,$$

$$\bar{w}_T^k(a) := \max_{\tau\in[\tau_{k-1},\tau_k]} w_T(a,\tau),$$

$$\bar{v}_k^G(a) := \sqrt{(v^A)^2 - \underline{w}_C^k(a)^2} + \bar{w}_T^k(a),$$

$$\bar{v}^G(a) := \max_{k\in\{1,\dots,K\}} \bar{v}_k^G(a),$$

and $\underline{T}(a) := \dfrac{d^G(a)}{\bar{v}^G(a)}$.

By definition, we know that on any time interval, the ground speed increases as the tailwind increases, and decreases as the crosswind increases. Thus, $(\underline{w}_C^k(a), \bar{w}_T^k(a))$ corresponds to an imaginary *super-optimal wind* vector whose corresponding ground speed $\bar{v}_k^G(a)$ overestimates the ground speed in the time interval $[\tau_{k-1}, \tau_k]$.

For an example, see Figure 3. On the right side, we see a typical behavior of the tail- and crosswind functions on an arc in a given time interval $[\tau_1, \tau_2]$, the minimum and maximum of interest are marked. On the left side, we see the super-optimal wind vector that results from the combination of both components. This vector yields a larger ground speed than all wind vectors in the gray rectangle, and thus larger than all wind vectors in the interval $[\tau_1, \tau_2]$, represented by the dashed curve.

From this overestimation property and the definition of $\bar{v}^G(a)$, it follows that, for each arc, the maximum of the ground speed overestimators on all discretization intervals overestimates the ground speed at any time, while the resulting travel time is a global underestimator:

▶ **Lemma 1.** *For every $a \in A$ and $\tau \in [t_0, t_N]$, we have*

$$\bar{v}^G(a) \geq v^G_*(a) \geq v^G(a,\tau) \quad \textit{and} \quad \underline{T}(a) \leq T^*(a) \leq T(a,\tau).$$

Thus, all we need to obtain the bounds $\bar{v}^G(a)$ and $\underline{T}(a)$ is to compute $\underline{w}^k_C(a)$ and $\bar{w}^k_T(a)$ for every $a \in A$, $k = 1, \ldots, K$. We will describe that step in Subsection 3.4. In the remainder of this subsection, we will prove that the absolute overestimation/underestimation error is linear with respect to the discretization step. Assuming that the aircraft is always at least twice as fast as the wind (which is always the case in practice), we can bound the constant in terms of the airspeed and the length of the weather prognosis intervals.

▶ **Theorem 2.** *For every $a \in A$, assume that $v^A \geq 2r^*_a$. Then, there exists a constant $C^v > 0$ such that the ground speed error is bounded as follows:*

$$0 \leq \bar{v}^G(a) - v^G_*(a) \leq C^v \Delta.$$

Proof. The left inequality follows from Lemma 1. For the right one we only have to prove that there exists $C^v > 0$ s.t.

$$\max_{\tau \in [\tau_{k-1}, \tau_k]} \big(\bar{v}^G_k(a) - v^G(a,\tau)\big) \leq C^v \Delta \quad \text{for every } k = 1, 2, \ldots, K. \tag{3}$$

To bound the ground speed error, we first bound the error on tailwind and crosswind. W.l.o.g assume $k = 1$ and define $I = [\tau_0, \tau_1]$. Let $\rho_1, \rho_2 \in I \subseteq [t_0, t_1]$ and $\lambda_1, \lambda_2 \in [0,1]$ satisfy $\rho_i = \lambda_i t_0 + (1-\lambda_i)t_1$, $i = 1, 2$. We have

$$\begin{aligned} |w_T(a,\rho_1) - w_T(a,\rho_2)| &\leq |\rho_1 - \rho_2| \max_{\rho \in I} |w'_T(a,\rho)| \\ &= |\rho_1 - \rho_2| \max_{\rho \in I} \Big| r'_a(\rho)\cos\big(\theta_a(\rho)\big) - r_a(\rho)\sin\big(\theta_a(\rho)\big)\theta'_a(\rho) \Big| \\ &\leq \Delta\big(\max_{\rho \in I} |r'_a(\rho)| + r^*_a \max_{\rho \in I} |\theta'_a(\rho)| \big), \end{aligned} \tag{4}$$

where $r^*_a = \max_{\rho \in [t_0, t_N]} r_a(\rho)$. Since wind speed and direction are interpolated linearly in $[t_0, t_1]$, we have

$$r'_a(\rho) = \frac{r^1_a - r^0_a}{t_1 - t_0} \quad \text{and} \quad \theta'_a(\rho) = \frac{\theta^1_a - \theta^0_a}{t_1 - t_0}. \tag{5}$$

From (4) and (5), it follows that

$$|w_T(a,\rho_1) - w_T(a,\rho_2)| \leq \Delta \frac{|r^1_a - r^0_a| + r^*_a|\theta^1_a - \theta^0_a|}{t_1 - t_0} \leq \Delta \frac{r^*_a(1+2\pi)}{t_1 - t_0}.$$

Similarly, we can prove that

$$|w_C(a,\rho_1) - w_C(a,\rho_2)| \leq \Delta \frac{r_a^*(1+2\pi)}{t_1 - t_0}.$$

We are now ready to establish a bound on the ground speed error. Let $\rho^*, \bar{\rho}, \underline{\rho} \in I$ satisfy $\rho^* \in \operatorname{argmax}_{\tau \in I} v^G(a,\tau)$, $w_T(a,\bar{\rho}) = \bar{w}_T^1(a)$, and $w_C(a,\underline{\rho}) = \underline{w}_C^1(a)$. The absolute ground speed error in interval I is thus

$$\begin{aligned}
\bar{v}_1^G(a) - v^G(a,\rho^*) &= \sqrt{(v^A)^2 - w_C(a,\underline{\rho})^2} + w_T(a,\bar{\rho}) - \sqrt{(v^A)^2 - w_C(a,\rho^*)^2} - w_T(a,\rho^*) \\
&= \frac{w_C(a,\rho^*)^2 - w_C(a,\underline{\rho})^2}{\sqrt{(v^A)^2 - w_C(a,\underline{\rho})^2} + \sqrt{(v^A)^2 - w_C(a,\rho^*)^2}} + w_T(a,\bar{\rho}) - w_T(a,\rho^*) \\
&\leq \frac{|w_C(a,\rho^*) - w_C(a,\underline{\rho})||w_C(a,\rho^*) + w_C(a,\underline{\rho})|}{\sqrt{(v^A)^2 - r_a(\underline{\rho})^2} + \sqrt{(v^A)^2 - r_a(\rho^*)^2}} + \Delta \frac{r_a^*(1+2\pi)}{t_1 - t_0} \\
&\leq \Delta \frac{r_a^*(1+2\pi)}{t_1 - t_0} \left(\frac{r_a(\rho^*) + r_a(\underline{\rho})}{2\sqrt{(v^A)^2 - {r_a^*}^2}} + 1 \right) \\
&\leq \Delta \frac{r_a^*(1+2\pi)}{t_1 - t_0} \left(\frac{r_a^*}{\sqrt{(v^A)^2 - {r_a^*}^2}} + 1 \right).
\end{aligned}$$

By assumption, the wind speed r_a^* is always smaller than half of the airspeed v^A, so we have

$$r_a^* \left(\frac{r_a^*}{\sqrt{(v^A)^2 - {r_a^*}^2}} + 1 \right) \leq \frac{v^A}{2}(1+1) = v^A.$$

In practice, r_a^* (wind speed) is usually much smaller than $\dfrac{v^A}{2}$ (flight speed), hence we can choose $C^v := \dfrac{v^A(1+2\pi)}{L}$. ◀

Using $\underline{T}(a)\bar{v}^G(a) = d^G(a) = T^*(a)v_*^G(a)$ and Theorem 2, the main result of this section follows:

▶ **Corollary 3.** *For every $a \in A$, assume that $v^A \geq 2r_a^*$. Then, there exists a constant C^T s.t.*

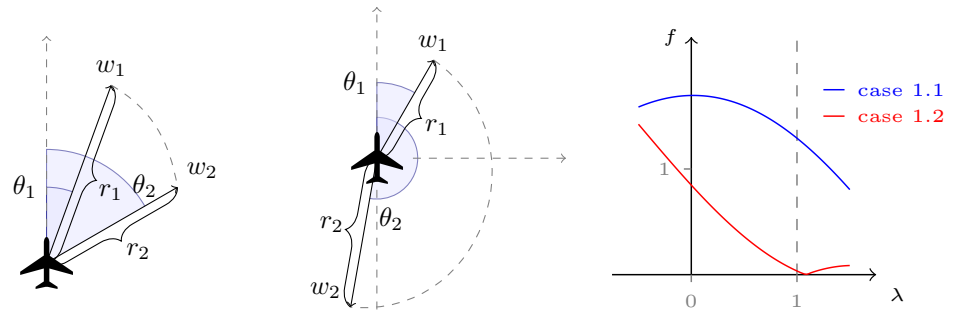
$$0 \leq T^*(a) - \underline{T}(a) \leq C^T \Delta.$$

That is, assuming reasonable wind conditions, the additive gap between the presented TTF underestimators and the corresponding minima is linearly bounded by the discretization step.

3.4 Minimization of Crosswind and Maximization of Tailwind

In the previous subsection, we used the minimum-magnitude crosswind in an interval in order to compute the super-optimal wind vector that is needed to define $\underline{T}(a)$. In this subsection, our objective is to show how this minimization can be done. We recall

$$|w_C(a,\tau)| = |\big(\lambda r_{k-1} + (1-\lambda)r_k\big) \sin\big((\lambda\theta_{k-1} + (1-\lambda)\theta_k\big)|,$$



(a) θ_1, θ_2 belong to the first quadrant.

(b) θ_1, θ_2 belong to different quadrants.

(c) Function f in case 1.

Figure 4 Cases considered for crosswind minimization.

where $\lambda := \frac{\tau_2 - \tau}{\tau_2 - \tau_1}$ and $\tau \in [\tau_{k-1}, \tau_k]$ for some $k = 1, \dots, K$. W.l.o.g., assume $k = 2$ for ease of notation. It suffices to consider the case $w_C(a, \tau) \geq 0$ for all $\tau \in [\tau_1, \tau_2]$. Indeed, if $w_C(a, \tau)$ takes both positive and negative values in $[\tau_1, \tau_2]$, by continuity the minimum absolute value must be 0, thus making the solution trivial. The case where $w_C(a, \tau) \leq 0$ for all $\tau \in [\tau_1, \tau_2]$ is analogous by symmetry. Thus, we can ignore the absolute values.

We can also assume that $\theta_1 < \theta_2$ and $r_1 \neq r_2$, as the other cases are either simple or can be reduced to this case. W.l.o.g. we will further assume that θ_1 and θ_2 belong to the same quadrant, since otherwise (see e.g. Figure 4b) we can compute the minimal value in each quadrant and take the overall minimum. Define

$$w_C(a, \tau) = \big(\lambda r_1 + (1-\lambda) r_2\big) \sin\big((\lambda\theta_1 + (1-\lambda)\theta_2\big) = (a\lambda + b)\sin(\alpha\lambda + \beta) =: f(\lambda)$$

with $a = r_1 - r_2$, $b = r_2 > 0$, $\alpha = \theta_1 - \theta_2 < 0$ and $\beta = \theta_2$. Its derivatives are then

$$\begin{aligned} f'(\lambda) &= a\sin(\alpha\lambda + \beta) + (a\lambda + b)\alpha\cos(\alpha\lambda + \beta), \\ f''(\lambda) &= 2a\alpha\cos(\alpha\lambda + \beta) - \alpha^2(a\lambda + b)\sin(\alpha\lambda + \beta), \\ f'''(\lambda) &= -3a\alpha^2\sin(\alpha\lambda + \beta) - \alpha^3(a\lambda + b)\cos(\alpha\lambda + \beta). \end{aligned}$$

We make the following case distinction:

1. $\theta_1, \theta_2 \in [0, \frac{\pi}{2}]$: We have $\lambda \in [0, 1]$, $\sin(\alpha\lambda + \beta), \cos(\alpha\lambda + \beta) \geq 0$. Consider the following two subcases (see Figures 4a and 4c):
   - 1.1. $a > 0$, i.e., $r_1 > r_2$: As $(a\lambda + b) > 0$ and since $\alpha < 0$ we have $f''(\lambda) < 0$ for all $\lambda \in [0, 1]$. Hence, f is concave and must attain its minimum at either 0 or 1.
   - 1.2. $a < 0$: Since $f'''(\lambda) > 0$, $f''(\lambda)$ is increasing. Evaluating f'' at $\lambda = 1$ results in two possibilities: If $f''(1) < 0$, we have that f is concave in $[0, 1]$, and hence its minimum must be attained at one of the boundary points. If $f''(1) > 0$, we further need to distinguish whether $f''(0) > 0$ (which means f is convex, see below) or $f''(0) < 0$. In the latter case, we perform a Newton procedure for finding the inflection point ($f''(\lambda) = 0$), and subdivide $[0, 1]$ into its convex and its concave part. Having done so, we know that the minimum in the concave part is attained at one of the end points. When f is convex, we apply Newton's method to find a root of $f'(\lambda)$. In case the minimum is found outside of $[0, 1]$, we simply take the $\lambda \in \{0, 1\}$ closest to it. Comparing the values from the concave and convex parts yields the minimum.
2. $\theta_1, \theta_2 \in [\frac{\pi}{2}, \pi]$: We have $\sin(\alpha\lambda + \beta) \geq 0$ and $\cos(\alpha\lambda + \beta) \leq 0$, and again distinguish between two subcases:

Table 1 Algorithm nomenclature used in the result tables.

Algorithm \ Problem	SPP	TDSPP-PWL	TDSPP-E
Dijkstra	DIJK	DIJK_{PWL}	DIJK_{E}
A*	A*	A^*_{PWL}	A^*_{E}
Contraction Hierarchies	CH	TCH	–

2.1. $a > 0$: Analogous to 1.2.
2.2. $a < 0$: Since $f''(\lambda) < 0$, analogous to 1.1.

3. $\theta_1, \theta_2 \in [\pi, \frac{3\pi}{2}]$: Analogous to 2 by symmetry.

4. $\theta_1, \theta_2 \in [\frac{3\pi}{2}, 2\pi]$: Analogous to 1 by symmetry.

Applying a very similar procedure to the function $g(\lambda)$ defined below yields the maximum of the tailwind component $w_T(a, \tau)$:

$$w_T(a,\tau) = \left(\lambda r_a^1 + (1-\lambda) r_a^2\right) \cos\left(\lambda\theta_a^1 + (1-\lambda)\theta_a^2\right) = (a\lambda + b)\cos(\alpha\lambda + \beta) =: g(\lambda).$$

3.5 Validation of Super-Optimal Wind Quality

To assess the quality of the super-optimal wind bounding procedure, we ran it on all arcs in nine instances, corresponding to the three graphs and three weather prognoses described below, in Section 4. Our weather prognoses satisfy $L = t_{i+1} - t_i$ equal to three hours for $i = 1, \ldots, n$. That, is, precise prognoses are given at three hour intervals and wind is interpolated for times in between. Thus, an obvious candidate for the discretization step Δ is at most three hours. Computational results show that our algorithm with $\Delta = L = 3\text{h}$ already provides excellent results. To validate our lower bounds, we also used a brute force approach which computes the maximal ground speed on each segment and each time interval through enumeration. The average relative error between our lower bounds and the brute force results is only $0.434 \cdot 10^{-3}$. Also, the average time it takes to process an arc is less than one millisecond; the average run time measured for the complete calculation is 5.61 seconds, running the code on 20 threads on the computer described in Section 4. Another interesting fact is that, in almost one third of the cases, the estimated result coincided with the exact result.

4 Computational Results

In this section, we present the results of extensive computations measuring the performance of our algorithms on airway networks.

For each of the considered problem variants (SPP, TDSPP-PWL, and TDSPP-E), we implemented a Dijkstra algorithm and an A* algorithm, using the potential functions described in Section 3. To test CHs and TCHs on our instances, we used the tools *Contraction Hierarchies* and *KaTCH*, both released by the Karlsruhe Institute of Technology (KIT) [15].

All algorithms (including the Contraction Hierarchies tools) were implemented in C++ and compiled with GCC, and all our computations were performed on computers with 132 GB of RAM and an Intel(R) Xeon(R) CPU E5-2660 v3 processor with 2.60GHz and 25.6 MB cache. All preprocessing steps were carried out in parallel using 20 threads, except for

Table 2 Graph instances corresponding to three common flight altitudes.

Instance	Nodes	Arcs	Avg. degree	Flight altitude
I-29	52719	329442	6.249	29000ft
I-34	52691	329736	6.258	34000ft
I-39	52662	329580	6.258	39000ft

Table 3 Comparison of CH and A* in the static case.

Instance	Dijk	CH			A*		
	query (ms)	prep (s)	query (ms)	speedup ×	prep (s)	query (ms)	speedup ×
I-29	2.01	1260	0.37	**5.45**	0	0.34	5.86
I-34	2.00	1233	0.38	5.24	0	0.33	**6.12**
I-39	1.94	1309	0.39	5.01	0	0.32	6.00

that of static Contraction Hierarchies, whose code does not offer the option of parallelization. All other computations were carried out in single-thread mode.

We use the notation introduced in Table 1 to refer to the different algorithms. In all subsequent tables, we use the abbreviations "prep" for preprocessing time, "query" for query time (given an OD pair) and "speedup" for the ratio between the given algorithm's query times and Dijkstra's query times.

4.1 Instances

All instances used in our computations correspond to real-world data, provided to us either by Lufthansa Systems GmbH & Co. KG (graphs and weather prognoses) or obtained from the flight tracking portal www.flightradar24.com (list of OD pairs).

We consider three directed graphs, corresponding to horizontal layers of the airway network at altitudes 29000 feet, 34000 feet, and 39000 feet, respectively. We chose these particular three because they are all common cruise altitudes distant enough from each other that the weather conditions are substantially different. While the graphs are topologically very similar, there exist several arcs which may be used only at certain altitudes. The characteristics of the three graphs and the notation we will use to refer to them are presented in Table 2.

Furthermore, we consider three different sets of weather prognoses. Each contains weather information for a period ranging from 30 to 45 hours, with prognoses available at intervals of 3 hours. We identify them by the names *Dec*, *Feb* and *Mar*, based on the dates in which the prognoses were made.

To construct instances for TDSPP-PWL, we approximated the TTFs with piecewise linear functions by discretizing the time horizon into time intervals of length one/three hours. Three hours is an obvious choice, since each prognosis set makes predictions for time points at three hours intervals. All TTFs thus obtained satisfy the FIFO property.

For all algorithms, we ran queries on a fixed set of 18644 OD pairs, corresponding to all flights recorded by www.flightradar24.com in June, 2015.

We use the following notation to identify our instances. For SPP, instances are given by the altitude (e.g. I-29). For TDSPP-E, instances are defined by the altitude and the weather prognosis set (e.g. I-29-Feb). Finally, for TDSPP-PWL, we identify instances by the altitude, prognosis, and discretization size (e.g. I-29-Feb-3).

Table 4 Comparison of TCHs and A^*_{PWL} for TDSPP-PWL.

Instance	$DIJK_{PWL}$	TCH			A^*_{PWL}		
	query (ms)	prep (min)	query (ms)	speedup ×	prep (s)	query (ms)	speedup ×
I-29-Dec-1	4.91	380.48	4.08	1.20	1.82	0.22	21.51
I-34-Dec-1	4.91	451.82	4.27	1.15	1.83	0.24	20.24
I-39-Dec-1	4.93	195.75	3.23	1.53	1.81	0.16	30.15
I-29-Feb-1	4.90	414.78	3.94	1.25	1.87	0.21	22.96
I-34-Feb-1	4.86	466.95	3.96	1.23	1.72	0.21	22.23
I-39-Feb-1	4.92	184.20	3.01	1.63	1.72	0.15	31.50
I-29-Mar-1	4.55	216.57	2.82	1.61	1.50	0.16	27.27
I-34-Mar-1	4.55	189.18	2.92	1.55	1.56	0.18	24.38
I-39-Mar-1	4.58	127.38	2.52	1.81	1.54	0.15	29.45
I-29-Dec-3	4.36	312.40	2.67	1.63	1.54	0.19	22.03
I-34-Dec-3	4.38	351.70	2.80	1.56	1.54	0.21	20.85
I-39-Dec-3	4.38	160.20	2.30	1.90	1.54	0.14	30.87
I-29-Feb-3	4.31	328.47	2.66	1.62	1.51	0.18	23.09
I-34-Feb-3	4.28	372.15	2.92	1.47	1.60	0.19	21.68
I-39-Feb-3	4.33	155.07	2.20	1.97	1.52	0.13	**31.94**
I-29-Mar-3	4.22	179.45	2.31	1.82	1.34	0.14	28.39
I-34-Mar-3	4.26	146.52	2.33	1.83	1.37	0.16	26.68
I-39-Mar-3	4.26	96.80	2.03	**2.10**	1.35	0.13	31.02
Summary							
Average	4.55	262.77	2.94	1.60	1.59	0.18	25.90
Minimum	4.22	96.8	2.03	1.15	1.34	0.13	20.24
Maximum	4.93	466.95	4.27	2.10	1.87	0.24	31.94

4.2 Static Case

The results for SPP can be found in Table 3. The speedup obtained by A* is slightly better than that of CHs, but not significantly. What is remarkable is that, since the potential for A* is computed on-the-fly during query time, no preprocessing is necessary. This results in a distinct advantage over CHs, which require over 20 min. preprocessing time. However, this is also the reason why the query times of A* are longer than in the time-dependent version (see Table 4). In fact, the computation of the potential functions accounts for over half the CPU time needed for the queries. The good performance of A* in this case is likely due to the fact that airway networks allow for minimum-distance paths to lie close to the great circle, as opposed to road networks.

4.3 Approximate Dynamic Case

In Table 4, we compare the results for the solution of TDSPP-PWL: A^*_{PWL} is the clear winner. We can see that the preprocessing time of TCHs is much longer than that of A*, and is in fact too long to be of use in practical applications. Furthermore, the query times of A^*_{PWL} yield an approximate speedup of 25 w.r.t. Dijkstra and 15 w.r.t. TCHs. Recall that A^*_{PWL} can exploit the fact that the set of possible target nodes is small and known in advance, while TCHs have no such advantage. This partially explains the former algorithm's superiority.

4.4 Exact Dynamic Case

Finally, in Table 5, we compare our versions of A* implemented for TDSPP-E and TDSPP-PWL. While $A^*_{PWL_1}$ and $A^*_{PWL_3}$ refer to the same algorithm, we use the indices 1 and 3 to distinguish between the instances with corresponding discretization steps. The preprocessing

Table 5 Comparison of A^*_E and A^*_{PWL} in the time-dependent case.

Instance	DIJK_E	A^*_E			$A^*_{PWL_1}$				$A^*_{PWL_3}$			
	query (ms)	prep (s)	query (ms)	speedup ×	prep (s)	av err (%)	max err (%)	bad paths (#)	prep (s)	av err (%)	max err (%)	bad paths (#(%))
I-29-Dec	100.89	7.51	5.80	17.38	139.41	0.059	**8.76**	506 (2.71%)	46.69	0.078	8.76	740 (3.97%)
I-34-Dec	102.12	7.38	6.13	16.64	140.81	0.072	5.30	701 (3.76%)	47.88	0.093	**10.92**	940 (5.04%)
I-39-Dec	104.33	7.56	4.47	23.34	140.98	0.018	2.65	79 (0.42%)	47.93	0.021	2.65	94 (0.50%)
I-29-Feb	100.88	7.66	5.49	18.37	139.74	0.028	5.38	195 (1.05%)	47.03	0.035	5.38	269 (1.44%)
I-34-Feb	101.37	7.35	5.68	17.85	141.32	0.038	4.64	317 (1.70%)	48.43	0.049	4.63	431 (2.31%)
I-39-Feb	104.44	7.45	4.16	**25.09**	140.72	0.015	3.60	51 (0.27%)	48.45	0.019	3.60	75 (0.40%)
I-29-Mar	100.07	7.14	4.85	20.60	91.38	0.022	5.37	96 (0.51%)	31.26	0.030	5.41	183 (0.98%)
I-34-Mar	35.72	5.77	1.85	19.25	92.78	0.019	4.60	87 (0.47%)	32.34	0.022	4.60	111 (0.60%)
I-39-Mar	36.18	5.68	1.59	22.66	95.21	0.016	4.74	89 (0.48%)	33.01	0.017	4.74	93 (0.50%)
Summary												
Average	87.33	7.06	4,45	20.13	124.71	0.032	5.00	235.67 (1.26%)	42.56	0.040	5.63	326.22 (1.75%)
Minimum	35.72	5.68	1,59	16.64	91.38	0.015	2.65	51.00 (0.27%)	31.26	0.017	2.65	75.00 (0.40%)
Maximum	104.44	7.66	6,13	25.09	141.32	0.072	8.76	701.00 (3.76%)	48.45	0.093	10.92	940.00 (5.04%)

times measured for $A^*_{PWL_1}$ and $A^*_{PWL_3}$ include both the construction of the piecewise linear functions (not considered in Table 4, since in that case the procedure is needed by all algorithms) and of the potential functions. Comparing the query times with those of $A^*_{PWL_1}$ and $A^*_{PWL_3}$ (Table 4) shows that the running time increases by a factor of over 20. This is a disadvantage of the exact method, even though it is still very fast.

On the other hand, measuring the impact of the approximations on the final solution reveals some outliers. To this purpose, we compare the optimal solution returned by Dijkstra with that returned by the A^*_{PWL} algorithms. For both solutions we compute the exact minimal travel time and the PWL objective value. Table 5 displays the average relative error, the maximum relative error, and the number of "bad paths", which are defined as OD pairs for which the relative error is larger than 0.5%. This value is interesting since, in the flight planning industry, savings of 0.5% can justify longer running times. The number of bad paths is not insignificant, and justifies the consideration of the exact method.

5 Conclusion

This paper shows that airway networks allow significant speedups in shortest path computations over Dijkstra's algorithm, but with different methods than those used for road networks. In particular, it turns out that the A* algorithm with problem-specific potentials performs better than Contraction Hierarchies. We discuss three different versions of the Horizontal Flight Planning Problem: The shortest path problem with static costs, the time-dependent shortest path problem with piecewise-linear TTFs, and a special case of the time-dependent shortest path problem with non-piecewise-linear (weather-dependent) TTFs.

For the first two variants, A* potentials based on GCDs and PWL approximations, respectively, are faster than CHs and TCHs. In both cases, the preprocessing time needed by A* is shorter than that of CHs by several orders of magnitude. In the static case, the query times of both algorithms are comparable, while in the case of piecewise linear TTFs, A* outperforms TCHs by a factor of 15. It remains an open question whether Contraction Hierarchies can be adapted to attain a better performance on airway networks.

For the variant of non-piecewise-linear TTFs, we propose a *super-optimal wind* procedure for underestimating TTFs. We present tight theoretical and empirical bounds on its approximation error. The A* algorithm resulting from these bounds yields a speedup factor of 20 with respect to Dijkstra and very short preprocessing times. We also analyze the effect of approximating TTFs with piecewise linear functions. This approximation approach leads to extremely fast query times and a very small average error, but produces a few outliers. An interesting research direction is to combine the advantages of these methods.

Future research also includes adapting these techniques for the three-dimensional flight planning problem. This is not straightforward since the TTFs corresponding to climb and descent phases depend not only on the wind, but also on the current aircraft's weight and technical specifications.

References

1. Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck. *Algorithms – ESA 2012: 20th Annual European Symposium, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, chapter Hierarchical Hub Labelings for Shortest Paths, pages 24–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
2. Hannah Bast, Daniel Delling, Andrew V. Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck. Route planning in transportation networks. *CoRR*, abs/1504.05140, 2015.
3. G. Veit Batz, Robert Geisberger, Peter Sanders, and Christian Vetter. Minimum time-dependent travel times with contraction hierarchies. *J. Exp. Algorithmics*, 18:1.4:1.1–1.4:1.43, April 2013.
4. Reinhard Bauer, Daniel Delling, Peter Sanders, Dennis Schieferdecker, Dominik Schultes, and Dorothea Wagner. Combining hierarchical and goal-directed speed-up techniques for dijkstra's algorithm. *J. Exp. Algorithmics*, 15:2.3:2.1–2.3:2.31, March 2010.
5. Pierre Bonami, Alberto Olivares, Manuel Soler, and Ernesto Staffetti. Multiphase mixed-integer optimal control approach to aircraft trajectory optimization. *Journal of Guidance, Control, and Dynamics*, 36(5):1267–1277, July 2013.
6. H.M. de Jong. Optimal track selection and 3-dimensional flight planning. Technical Report 93, Royal Netherlands Meteorological Institute, 1974.
7. Daniel Delling, Andrew V. Goldberg, Thomas Pajor, and Renato F. Werneck. Robust exact distance queries on massive networks. Technical Report MSR-TR-2014-12, July 2014.
8. Daniel Delling and Dorothea Wagner. *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*, chapter Time-Dependent Route Planning, pages 207–230. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
9. EUROCONTROL. Route availability document, 2016. [Online; accessed 2-June-2016]. URL: `https://www.nm.eurocontrol.int/RAD/`.
10. Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. Exact routing in large road networks using contraction hierarchies. *Transportation Science*, 46(3):388–404, August 2012.
11. A. V. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. Technical Report MSR-TR-2004-24, Microsoft Research, Vancouver, Canada, July 2004.
12. P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.
13. Stefan E. Karisch, Stephen S. Altus, Goran Stojković, and Mirela Stojković. Operations. In Cynthia Barnhart and Barry Smith, editors, *Quantitative Problem Solving Methods in the Airline Industry*, volume 169 of *International Series in Operations Research & Management Science*, pages 283–383. Springer US, 2012.
14. Giacomo Nannicini, Daniel Delling, Dominik Schultes, and Leo Liberti. Bidirectional a* search on time-dependent road networks. *Netw.*, 59(2):240–251, March 2012.
15. Peter Sanders, Moritz Kobitzsch, Veit Batz, Robert Geisberger, Dennis Luxen, Dennis Schieferdecker, Dominik Schultes, and Christian Vetter. Fast and exact route planning, 2016. [Online; accessed 2-June-2016]. URL: `http://algo2.iti.kit.edu/routeplanning.php`.