Dynamic Programming for Vertical Flight Planning

Z. Yuan and É. Taillard

IRIDIA – Technical Report Series
TR/IRIDIA/2016-006
May 2016
Last revision: June 2016
Dynamic Programming for Vertical Flight Planning

Zhi Yuan\(^1\) and Éric Taillard\(^2\)

\(^1\) IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
\(^2\) HEIG-VD, University of Applied Sciences of Western Switzerland, Yverdon-les-Bains, Switzerland

Abstract. Vertical flight planning concerns assigning optimal cruise altitude and speed to a flight trajectory, such that the fuel consumption is minimized, and the arrival time constraints are satisfied. The previous work mainly focused on using mixed integer programming solvers for this task. Here we present another exact approach based on dynamic programming. Its computational effectiveness is validated by various benchmark instances of realistic sizes.

Keywords: Flight planning; dynamic programming; exact algorithm; bi-objective; variable discretization

1 Introduction

The flight planning problem concerns finding a fuel-optimal trajectory in a four-dimensional (4D) space, including a 2D space on the surface of the earth, a vertical dimension consisting of discrete flight levels, and a time dimension of time constraints which can be controlled by aircraft speed. Due to the computational difficulty of this 4D optimization problem, it is common in practice [3] and in academia [17] to divide it into two subproblems: the first horizontal planning phase searches for a trajectory connecting the departure and the destination on the earth surface, and each segment composing the horizontal trajectory is assigned with a flight level and an aircraft speed in the subsequent vertical planning phase, such that the fuel consumption is minimized, and the time constraints are satisfied. A mixed integer programming (MIP) model for the horizontal planning in the context of free-flight is presented in [17].

In this work, we focus on the vertical flight planning phase, which can be applied to both free-flight [1, 17] or the conventional network-based flight planning [3, 13]. The vertical flight planning has been formulated as mixed integer second-order cone programming model when continuous speed values are used [18, 17]. If only discrete speed values are used, the model can be reformulated as a mixed integer linear programming model, and it significantly accelerated the solution approach without compromising the objective loss due to discretization error [16]. In this work, we follow the use of discrete speed [16], develop a dynamic programming approach, and examine its scalability on the flight planning problems of realistic sizes.
2 Literature review

The flight planning problem is a four-dimensional (4D) optimization problem. There exists work that formulates it as an optimal control problem, and computes a smooth 4D flight trajectory [14]. However, such a theoretical 4D curve is not feasible in practice due to air traffic control restrictions, e.g., only a discrete set of admissible flight altitudes are allowed, and an aircraft is only allowed to change altitude, direction, and speed at certain validated way points. Most of the approaches in practice [3] as well as in academia [6, 10, 17] solve this 4D problem with two phases: a horizontal phase that optimizes a 2D trajectory, and a vertical phase that assigns altitude and speed to the trajectory.

This work focuses on the vertical planning phase. The vertical profile of a flight consists of five stages: take-off, climb, cruise, descend, and landing. We focus on the cruise stage, since it consumes most of the fuel and time during a flight, while the other stages are relatively short and usually have fixed procedures due to safety considerations, thus leave little flexibility for fuel optimization. Computing an optimal altitude profile without considering wind can also provide estimated altitude for the 2D horizontal trajectory optimization [10]. Such a steady-atmosphere optimal altitude profile increases approximately linearly as fuel burns, however, it becomes irregular and multi-modal if altitude-dependent wind is considered [8]. Most of the existing optimal control approaches for optimizing the altitude and speed profile assume that a smooth and differentiable fuel consumption function is given [7, 14, 10]. This does not correspond to the flight planning practice, where the fuel consumption function as provided by the aircraft manufacturers are tabulated data, and the real consumption value need to interpolated by adjacent tabulated values. There exist approaches that use a global polynomial function (of usually degree 5) to approximate the fuel consumption table [11, 4] or even the wind data [7]. However, such global approximation approaches may lead to accuracy loss, and the estimation accuracy is known to be the most important concern in flight planning.

A work by Lovegren and Hansman [9] confirmed a potential fuel saving of up to 3.5% by reassigning only altitude and speed to fixed flight trajectories, based on a study of 257 real flight operations in US. However, no time constraint is taken into account in their computation as in real-world airline operations. In such case, there exists an exact backward dynamic programming approach to compute fuel-optimal vertical profile [17].

A practical challenge in airline operations is to handle time constraints, especially delays, due to disruptions such as undesirable weather conditions, unexpected maintenance requirements, or waiting for passengers or crew members transferring from other already delayed flights. Such delays are typically recovered by increasing cruise speed, such that the next connection for passengers as well as for the aircraft and the crew can be reached [2]. Varying cruise speed may also be useful, e.g., to enter a time-dependent restricted airspace before it is closed (or after it is open), or when an aircraft is reassigned to a flight that used to be served by a faster (or slower) aircraft. The industrial standard suggests using a cost index procedure to vary cruise speed. This requires inputing a value that
reflects the importance between time-related cost and fuel-related cost. The use of cost index was criticized due to the difficulty to quantify the time-related cost in the presence of delay, thus a dynamic cost index approach has been proposed to this end [5]. However, such approach still cannot handle explicitly hard time constraints, such as the about-to-close airspace. Hagelauer and Mora-Camino [6] considered explicit time constraints in the vertical flight planning, and proposed a heuristic based soft dynamic programming approach. However, their approach cannot guarantee global optimality; besides, their experiment also did not take wind influence into account. Aktürk et al. [2] formulate the time constraint explicitly into a MIP model in the context of aircraft rescheduling. However, their mathematical model only considered assigning a constant cruise speed for the whole flight. Yuan et al. [18, 17, 16] explicitly include the time constraint and the use of variable speed in the vertical flight planning.

3 Vertical Flight Planning: The Mathematical Model

3.1 Vertical Flight Planning (VFP) without Wind

The vertical flight planning (VFP) usually serves as a second step of the whole flight planning problem [3, 17], after a set of flight segments $S$ that compose the trajectory have been determined. The task is to assign speed and altitude to each segment $i \in S := \{1, \ldots, n\}$, such that the fuel consumption is minimized, and the arrival time is within a prescribed time window $[T, \bar{T}]$. The aircraft performance data for cruise provided by the aircraft manufacturer represent the unit distance fuel consumption depending on three factors: aircraft weight, flight altitude, and speed. Each of the three factors is divided into a number of discrete values, denoted $W$ for the weight grids, $H$ for altitude, and $V$ for speed. For computing the unit distance fuel consumption for a combination of these three factors outside the prescribed grid vertices, their value need to be interpolated by the adjacent grid values. If wind influence is not considered [18, 16], we can precompute the fuel consumption $F_{i,v}(w)$ for a segment $i \in S$ with length $L_i$ with a speed assignment $v \in V$ for each weight grid point $w \in W$ by enumerating the all the altitudes $H$ and picking the one with minimal fuel consumption. Besides, we can also precompute the flight time $\Delta T_{i,v}$ segment $i \in S$ by assigning speed $v \in V$. The problem is to assign discrete speed levels $v \in V$ to each segment $i \in S$, such that sum of flight time $t_i$ on each segment $i \in S$ is within a time window, and the sum of the fuel consumption is minimized. The MIP model of the VFP using discrete speed [16] is as follows:

$$\min \quad w_0 - w_n$$

s.t. \hspace{1cm} $t_0 = 0, \quad T \leq t_n \leq \bar{T}$ \hspace{1cm} (2)

$\forall i \in S : \quad \Delta t_i = t_i - t_{i-1}$ \hspace{1cm} (3)

$\forall i \in S : \quad \sum_{v \in V} \mu_{i,v} = 1$ \hspace{1cm} (4)
\[ \forall i \in S : \quad \Delta t = \sum_{v \in V} \mu_{i,v} \cdot \Delta T_{i,v} \quad (5) \]

\[ \omega_n = W_{empty} \quad (6) \]

\[ \forall i \in S : \quad \omega_{i-1} = \omega_i + f_i \quad (7) \]

\[ \forall i \in S : \quad f_i = \sum_{v \in V} \mu_{i,v} \cdot \hat{F}_{i,v}(\omega_i). \quad (8) \]

The total fuel consumption to be minimized (1) is measured by the difference of aircraft weight before and after the flight, \( w_0 \) and \( w_n \), respectively, where \( n := |S| \) is the number of segments; (2) initializes time and ensures that the flight duration is within a given time window; (3) preserves the time consistency by defining the travel time \( \Delta t_i \) on each segment \( i \in S \); Only one speed \( v \) is assigned to each segment \( i \) in (4) by introducing binary variables \( \mu_{i,v} \), and the travel time on each segment depends on the speed assignment by (5). (6) initializes the weight vector by assuming all trip fuel is burnt during the flight, and \( W_{empty} \) denotes the weight of the aircraft with an empty tank for trip fuel, i.e., the dry weight of the aircraft plus all the loads such as passengers, luggages, and different types of contingency fuel; weight consistency is ensured in (7) and the fuel consumption of each segment in (8) is calculated based on the speed selection \( \mu \) and a piecewise linear function \( \hat{F}_{i,v} : [W_{empty}, W_{max}] \to \mathbb{R} \) interpolating the fuel consumption \( F_{i,v} \) on segment \( i \) with speed \( v \) based on a continuous value of weight. The piecewise linear function can be modelled by various different formulation techniques, please see [16, 15] for details.

### 3.2 Vertical Flight Planning with Wind (VFPW)

The vertical flight planning with wind (VFPW) is first presented in [17] with continuous speed, and its speed discretization scheme is presented in [16]. Since wind is altitude dependent, which influences the segment fuel consumption as well as the segment travel time, not only speed but also altitude will be assigned to each segment. The binary variables \( \mu_{i,v} \) in (5) are extended to \( \mu_{i,v,h} \) for selecting a speed \( v \in V \) and an altitude \( h \in H \) for a segment \( i \in S \), and (4) is replaced by

\[ \forall i \in S : \quad \sum_{h \in H, v \in V} \mu_{i,h,v} = 1, \quad (4') \]

to ensure exactly one speed and one altitude is assigned to each segment. Accordingly, the travel time \( \Delta T_{i,v} \) is extended to \( \Delta T_{i,v,h} \), which is precomputed as travel time on segment \( i \in S \) assigned with speed \( v \in V \) and altitude \( h \in H \); similarly, \( F_{i,v,h} \) is precomputed as the fuel consumption on a segment \( i \) with speed \( v \) and altitude \( h \). Then the equations for travel time (5) and fuel consumption
Algorithm 1 DP-uVFP(W) \((S, V, (H,) F_{i,v,h})\)

1: \(w_n \leftarrow \text{wempty}\) \(\triangleright\) assume to arrive with all fuel consumed
2: \(l^n_n \leftarrow \emptyset\) \(\triangleright\) initialize optimal partial solution
3: for \(i \in \{n, \ldots, 1\}\) do
4: \(v^*_i, h^*_i \leftarrow \arg \min_{v,h \in V \times H} F_{i,v,h}(w_i)\) \(\triangleright\) select fuel-optimal speed and altitude
5: \(l^n_{i-1} \leftarrow l^n_i \oplus \{v^*_i, h^*_i\}\) \(\triangleright\) append it to optimal partial solution
6: \(w_{n-i} \leftarrow w_i + F_{i,v^*_i,h^*_i}(w_i)\) \(\triangleright\) update weight for the start of the segment
7: end for
8: return the optimal assignment \(l^n_0\)

(8) can be replaced by:

\[\forall i \in S : \Delta t = \sum_{h \in H, v \in V} \mu_{i,h,v} \cdot \Delta T_{i,h,v}, \quad (5')\]

\[\forall i \in S : f_i = \sum_{h \in H, v \in V} \mu_{i,h,v} \cdot \hat{F}_{i,h,v}(w_i). \quad (8')\]

To sum up, the MIP model for VFPW is as follows:

\[
\text{VFPW} \ (S, V, H, F_{i,v,h}, \Delta T_{i,v,h}): \\
\text{minimize (1)} \\
\text{subject to (2), (3), (4'), (5'), (6), (7), (8').} \\
\forall i \in S \cup \{0\} : w_i, t_i \in \mathbb{R}; \forall i \in S : \Delta t_i, f_i \in \mathbb{R}; \forall i \in S, v \in V, h \in H : \mu_{i,v,h} \in \{0, 1\}.
\]

4 The Dynamic Programming Approach

The vertical flight planning problems described in Section 3 were solved in the literature [18, 17, 16] using mixed integer programming (MIP) approaches. In this work, we explored the dynamic programming approach for them.

4.1 Dynamic Programming for unconstrained VFP(W)

If the arrival time constraint (2) in VFP and VFPW is not considered, such a unconstrained vertical flight planning problem (uVFP) can be solved by a backward dynamic programming approach as described in [17]. This approach, dubbed DP-uVFP(W) is outlined in Algorithm 1. Since the landing weight of the aircraft is known (initialized in line 1), we assign optimal speed and altitude levels segment by segment in the backward order, from the last segment to the first (line 3). For each segment \(i\), we select the best speed and altitude for it
Algorithm 2 DP-VFP ($S, V, F_{i,v}, ΔT_{i,v}$)

1: $w_n ← W_{empty}$ \Comment{assume to arrive with all fuel consumed}
2: preprocess weight bounds
3: preprocess time bounds
4: List $L_i^* ← \emptyset$ for all $i \in \{0, 1, \ldots, n\}$ \Comment{initialize the list of Pareto partial solutions}
5: for $i \in \{n, \ldots, 1\}$ do \Comment{loop all segments in reversed order}
6: for $l ∈ L_i^*$ do \Comment{loop the list of all Pareto partial solutions}
7: determine adjacent weight grid indices for $l$
8: for $v ∈ V$ do \Comment{loop all speed levels}
9: $L_i ← L_i \cup \{assignSpeedToOneSegment(l, v, i)\}$
10: end for
11: $L_i^* ← checkDominance(L_i^*)$ \Comment{remove the dominated partial solutions}
12: end for
13: return an assignment $l^* ∈ L_0 : l^* := \arg \max_{i \leq n} T_i \subseteq [\underline{T}, \overline{T}]$

According to the segment end weight $w_i$ (in line 4). Firstly, the adjacent weight grid indices for fuel interpolation $F(w)$ can be found in $O(\log_2 |W|)$ time by a binary search in the weight grid $W$. Then the best speed and altitude assignment that consumes minimal fuel is determined by enumerating all the discrete speed and altitude levels at a time complexity of $O(|V| \cdot |H|)$. The selected speed and altitude is appended to the optimal partial solution $l_i^*$ in line 5 to $l_{i-1}^*$, and the weight is updated accordingly in line 6. The partial solution $l_{i-1}^*$ is indeed optimal, since smaller end weight always leads to less fuel consumption on the segment, thus the Bellman’s principle of optimality holds. The time complexity of algorithm DP-uVFPW is $O(n \cdot (\log_2 |W| + |V| \cdot |H|)) ≈ O(n \cdot |V| \cdot |H|)$, since $\log_2 |W| \ll |V| \cdot |H|$, besides, one can further speed up the search of weight grid indices by starting from the weight indices of the previous segment. The time complexity of DP-uVFP is $O(n \cdot |V|)$.

4.2 Dynamic Programming for VFP

Algorithm 1 described in Section 4.1 computes the fuel-optimal vertical profile without time constraint. If a time constraint is imposed which does not allow the unconstrained optimal time, e.g., a speedup of the aircraft is required to mitigate delays, the partial solutions $l^*$ in Algorithm 1 that are fuel-optimal in the last segments will not guarantee global optimum since the unassigned segments may need to be assigned with much larger speed. Our optimal partial solutions have to take into account also the flight time besides the fuel consumption. That is, the optimal partial solution at each segment $i$ is in fact a list $L_i^*$ of Pareto optimal partial solutions in terms of flight time and fuel consumption. The DP approach for VFP, DP-VFP, is outlined in Algorithm 2.

The DP approach for VFP starts from the last segment $n$, with the known ending weight $w_n = W_{empty}$ in line 1. Then it assigns speed segment by segment from the last segment $n$ to the first segment (in line 5) iteratively. At each
Procedure 3 assignSpeedToOneSegment \((l, v, i)\), to assign speed \(v\) to a partial solution \(l\) on segment \(i\)
\[
\begin{align*}
    l & \leftarrow l \oplus v \\
    l.time & \leftarrow l.time + \Delta T_{i,v} \\
    l.weight & \leftarrow l.weight + \hat{F}_{i,v}(l.weight)
\end{align*}
\]
return extended partial solution \(l\)

Procedure 4 checkDominance \((L)\), to remove the dominated partial solutions from the list \(L\) in terms of time and weight.
\[
L \leftarrow \text{sort } L \text{ in ascending order of primarily time, and secondarily weight}
\]
\begin{algorithmic}
    \State loop \(l \in L\) in ascending order
        \State remove \(l\) from \(L\) if its weight is not smaller than the weight of the previous non-dominated partial solution
    \Endloop
\end{algorithmic}
return a Pareto optimal list \(L\) of partial solutions.

As a speed level \(v\) is assigned to a segment \(i\) for a partial solution \(l\), as outlined in Procedure 3, the assigned speed is appended, and both the flight time and aircraft weight are updated according to the speed assignment. Then the total number of partial solutions in \(|L^*_i|\) is equal to \(|L^*_i| \cdot |V|\). A further Procedure 4 is performed to reduce the number of the Pareto partial solutions by removing the partial solutions that are dominated by at least one other partial solution in terms of flight time and fuel consumption. We call a solution \(l_1\) is dominated by another \(l_2\) in the bi-objective sense if \(l_1\) is strictly worse than \(l_2\) in one of the objectives, and not better in the other objective. Such dominance checking procedure first sort the list \(L\) in ascending order of primarily flight time, and secondarily weight, and then traverse the sorted list to remove solutions whose weight is not smaller than the previous non-dominated solution. The time complexity from line 6 to line 11 is \(O(|L| \cdot (\log_2 |W| + |V|))\), where \(O(\log_2 |W|)\) complexity is needed for determining the adjacent weight grid indices of a partial solution \(l\) in line 7 for fuel interpolation; and line 12 requires \(O(|L| \cdot \log_2 |L|)\) with a Timsort [12] applied in line 1 of Procedure 4, which is a more sophisticated version of merge sort. Then the total time complexity of DP-VFP is \(O(n \cdot |L| \cdot (\log_2 |L| + \log_2 |W| + |V|))\). The number of Pareto partial solutions \(|L|\) can be bounded by, e.g., a time-discretization scheme explained in Section 4.5.

### 4.3 Dynamic programming for vertical flight planning with wind

In the vertical flight planning with wind (VFPW) described in Section 3.2, not only speed but also an altitude level should be assigned to each segment. The DP approach for VFPW, DP-VFPW is outlined in Algorithm 5. In comparison to DP-VFP, it will loop the speed-altitude pairs including also altitude levels \(H\) in line 9. This makes the time complexity of DP-VFPW to be \(O(n \cdot |L| \cdot (\log_2 |L| + |V|))\).
Algorithm 5 DP-VFPW (S, V, H, F_{i,v,h}, ΔT_{i,v,h})

1: \( w_n \leftarrow w^{empty} \) \triangleright assume to arrive with all fuel consumed
2: preprocess weight bounds
3: preprocess time bounds
4: preprocess speed-altitude pairs \( P_i \subseteq V \times H \)
5: List \( L_i^* \leftarrow \emptyset \) for all \( i \in \{0, 1, \ldots, n\} \) \triangleright loop all segments in reversed order
6: for \( i \in \{n, \ldots, 1\} \) do \triangleright loop the list of Pareto partial solutions
7: for \( l \in L_i^* \) do \triangleright loop all speed-altitude pairs
8: determine adjacent weight grid indices for \( l \)
9: for \( v,h \in P_i \) do \triangleright loop all speed-altitude pairs
10: \( L_i \leftarrow L_i \cup \{ \text{assignSpeedAltitudeToOneSegment}(l, v, h, i) \} \)
11: end for
12: end for
13: \( L_i^* \leftarrow \text{checkDominance}(L_i^*) \) \triangleright remove the dominated partial solutions
14: end for
15: return an assignment \( l' \in L_0 : l' := \arg \max_{i \in \mathbb{L}_0} T_i \subseteq [T, T] \)

\[ \log_2 |W| + |V| \cdot |H| \]. As we will see in Section 4.4, not all the speed-altitude pairs in \( V \times H \) need to be assigned on each segment, and some speed-altitude pairs that are dominated by another pair can be eliminated in the preprocessing phase to speed up the algorithm.

4.4 Preprocessing for Dynamic Programming Approaches

The DP approaches can be further speeded up by restraining the search within a certain range of weight, time, and certain combination of speed-altitude pairs. The task of the preprocessing is to reduce the solution space without excluding the global optimal solution. Three preprocessing steps are detailed below: preprocessing for weight, time, and speed-altitude pairs.

Preprocessing for Weight Bound The preprocessing for the weight bound has two advantages: it reduces the size of the weight grid \( |W| \); and it can also prune part of the solution space. The process can be illustrated in Figure 1. We first start the backward weight bound calculation from the end of the last segment \( n \) with the ending weight \( W^{empty} \). Then we enumerate all the assignment possibilities for the last segment and calculate the minimum and maximum possible weight \( w_n^{\text{min}} \) and \( w_n^{\text{max}} \) for the last segment, so that the starting weight of the last segment \( w_{n-1} \) must be within the range of \([W^{empty}+w_n^{\text{min}}, W^{empty}+w_n^{\text{max}}]\).

We iterate such process for each segment from the last to the first and sum up the either the largest or the smallest possible weight for each segment as upper or lower bound on this segment, respectively. The weight bound from this backward calculation is illustrated in Figure 1 with the solid blue line. In the second phase, we calculate from the start of the first segment \( 0 \) with the maximum take-off weight \( W^{\text{max}} \) of the aircraft, and iteratively compute the upper bound for the weight by subtracting from the previous weight value the minimum weight.
Dynamic Programming for Vertical Flight Planning

Fig. 1. The preprocessing of weight bounds. The gray grid represents the weight grid on each segment, the two solid blue lines represent the weight bounds calculated backwards from $W_{empty}$ at the last segment, and the solid green line represents the weight bound calculated from the maximum weight at take-off. The weight grid in gray is reduced to the area marked in yellow, and the solution space in red is pruned.

Preprocessing for Weight Bound

on the next segments. The new upper bound by the forward calculation is illustrated in Figure 1 by the green line. Then feasible solutions are within the yellow region, which significantly reduced the number of weight grid points as marked in gray. Besides, once a partial solution during the later DP approach reach the red region, it will be discarded from the Pareto list, since it will definitely lead to an over-weighted aircraft at take-off. If a feasible solution is already known, this weight preprocessing can be extended by replacing $W_{max}$ by the best-known fuel consumption, such that the solution space can be further reduced. Besides, if a reasonable lower bound of the total fuel consumption exists, it could be also used to provide a lower bound in the forward calculation phase to prune the partial solutions that consume unrealistically little fuel in the early construction phase.

Preprocessing for Time Bound

We assume that the flight time at the end of segment $n$ is initialized to 0, and the flight time at each segment is summed up from the last segment to the first segment, then the time at the start of the first segment, i.e., 0 at the $x$-axis, represents the total flight time. The time bounds in the backward calculation is illustrated by the two solid blue lines in Figure 2, with the upper bound and lower bound being the assignments with always the maximum and minimum flight time on each segment, respectively. Since we consider here the flight plans that require shorter time than its unconstrained optimal flight time, the fuel-optimal solution should be as close as possible to the upper flight time bound $T$. We allow solutions with total flight time between $T - \epsilon$ and $T$. The time deviation $\epsilon$ is to ensure that the optimal solution with discrete
Fig. 2. The preprocessing of time bounds. The two solid blue lines represent the original time bounds calculated backwards that all the partial solutions without time constraint lie. The solid green lines represent the time bound calculated forwards from the maximum arrival time (with an $\epsilon$ due to discrete speed assignment). The solution space in yellow is to be searched by DP, and the solution space in red is pruned.

speed assignment is not excluded. Then we start the time bound calculation forward from the time bound $[T - \epsilon, T]$ at segment index 0, and compute the lower bound and the upper bound for finishing the first segment at segment index 1 by subtracting the maximum and minimum flight time from its previous lower and upper bound, i.e., $[T - \epsilon - t_{1}^{\text{max}}, T - t_{1}^{\text{min}}]$, respectively. This process iterates forward to the next segments, and the resulting time bound is marked by two green lines in Figure 2. The solution space is significantly reduced to the yellow region in Figure 2, and the partial solutions in the red region either flies too slow or too fast in the early construction phase to become the optimal solution, thus will be discarded immediately from the Pareto partial solution list.

Preprocessing for Speed-Altitude Pairs The DP approach for VFPW (DP-VFPW) in Algorithm 5 assigns speed-altitude pairs $P_i$ at each segment $i$ to each previous Pareto partial solution in line 9. The number of speed-altitude pairs at each segment has great impact on the computational complexity. A speed-altitude pair $p_1$ can be discarded in preprocessing, if there exists another pair $p_2$ that dominates it, i.e., when $p_1$ is dominated by $p_2$ in terms of flight time and fuel consumption across all the possible weight values. Note that with the same speed and weight, if one altitude level is shorter in time than the other, it implies that the wind helps more in the former case, but it does not imply that it will consume less fuel because altitude also plays a role in fuel consumption. Since the favoring altitude in fuel consumption depends on the weight, we compare each two pairs in $P_i$ for a segment $i$ on all the possible weight grid values within
the preprocessed weight bounds on segment $i$, and discard the pairs that are dominated by at least one other pair across all the weight grid values. This procedure usually eliminates a significant amount of dominated speed-altitude pairs, and speeds up the computation drastically.

4.5 Time Discretization in Dynamic Programming

The Pareto partial solution list $L_i^*$ may grow exponentially as the number of segments $i$ increases. In the worst case, the size $|L_i^*| = |V|^n$ for the VFP, and $|L_i^*| = |V|^n|W|^n$ for the VFPW. The number of partial solutions can be bounded, if the flight time at each segment as shown in the yellow region in Figure 2 is discretized, such that the decision of dominance check in Procedure 4 is based on the flight time rounded to a discrete level. Denote $T_i$ the set of discrete time levels at a segment $i$, then the Pareto partial solution list size $|L_i^*| \leq |T_i|$. The coarser the time discretization, the less number of elements in $T_i$ will be, thus smaller the $|L_i^*|$, and faster the computation. However, coarser time discretization will also lead to suboptimal solutions, since the optimal solution may have been incorrectly discarded during the dominance check procedure due to rounding into a discrete level. We will experimentally examine the difference coarseness of the time discretization in terms of computation time and objective deviation from the exact optimal solution.

5 Experimental Results

5.1 Experimental Setup

Four of the most common aircraft types, Airbus 320, 380 (A320 and A380) and Boeing 737, 777 (B737 and B777), are used for the empirical studies in this work. The aircraft performance data and the upper air data are provided by Lufthansa Systems AG. Two vertical flight planning problems are considered, the one without wind (VFP), and the other with wind (VFPW). For VFP, the instances considered in [18] for continuous speed are adopted here, including two speedup factors (flying 2.5% and 5% faster than unconstrained optimal).

Five instances sizes are considered for A320, ranging from 15, 20, 25, 30, and 35 segments, each of which is 100 nautical miles (NM) long, which results in flight ranges from 1500 NM to 3500 NM; four B737 instance sizes are considered: 8, 12, 15, 18, i.e., flight ranges from 800 NM to 1800 NM; five A380 instance sizes with 40, 50, 60, 70, 75 segments, i.e., ranging from 4000 to 7500 NM; five B777 instances with 35 to 75 segments ranging from 3500 to 7500 NM; totalling 38 instances. The details of the instances are listed in Table 1.

For VFPW, we adopted the instances considered in [16], with two different wind fields (one for eastwards, and one for its westwards return trip), three speedup factors: 2%, 4%, and 6%, and three different instance sizes: 10, 20, and 40 segments of 75 NM each for A320, and 10, 15, 20 segments of 75 NM each for B737, totaling 36 instances. In addition, we also included the two large aircrafts
Table 1. Four aircraft types, Airbus 320, 380, Boeing 737, 777, and their characteristics, such as optimal and maximal speed (in Mach number), dry weight and maximal weight (in kg), and maximal distance (in NM). The number of speed grids $|V|$ (between optimal and maximal speed) and weight grids $|W|$, and the empirically determined conic approximation level $J$ are also listed.

| Type | Opt. Speed | Max. Speed | Dry Weight | Max. Weight | Max. Distance | $|V|$ | $|W|$ |
|------|------------|------------|------------|-------------|---------------|------|------|
| A320 | 0.76       | 0.82       | 56614      | 76990       | 3500          | 7    | 15   |
| A380 | 0.83       | 0.89       | 349750     | 569000      | 7500          | 7    | 24   |
| B737 | 0.70       | 0.76       | 43190      | 54000       | 1800          | 7    | 12   |
| B777 | 0.82       | 0.89       | 183240     | 294835      | 7500          | 8    | 16   |

A380 and B777 which is not solvable by the MIP solver in [16]. Three instance sizes are considered for the both large aircrafts, 40, 60, and 80 segments of 75 NM each.

All experiments ran on a computing node with a 12-core Intel Xeon X5675 CPU at 3.07 GHz and 48 GB RAM. The MIP solver reported here is Cplex 12.6, since Gurobi 6.0 leads to infeasible solutions with violated constraints. Each solver run uses 12 threads for the MIP solver, and an instance is considered optimally solved, when the MIP gap is within 0.01%, which corresponds to a maximum fuel error of 1 kg for B737, and maximum 2 kg for A320. The DP approach uses only a single thread, although its potential of parallelization is also promising to be explored in future works.

5.2 Computation Time Comparing with MIP Solver

Significant speedup of the MIP approach for the vertical flight planning has been achieved by different MIP formulation techniques including second-order cone linearization techniques [18, 17], various piecewise linear interpolation formulations [17, 16], and an automatic configuration of the state-of-the-art MIP solver [15]. The MIP approach can solve the VFP with the two small aircraft types A320 and B737 to optimality within seconds [16, 15], and VFPW with A320 and B737 to optimality within 2 minutes [16]. However, for the large instances with the larger aircraft types A380 and B777, VFP already cannot be solved to optimality for all instances, leaving for one instance an optimality gap of 0.1% after four hours’ runtime, as shown in Figure 3(a-b); for VFPW, most of the largest instances with 60 or 80 segments cannot be solved to optimality, leaving a gap of up to 2%, as shown in Figure 3(c-d). In contrast, the DP approach is much faster. It finds exact optimum for VFP for all benchmark instances within 0.02 seconds, as shown in Figure 3; In the computationally more challenging VFPW benchmark, the largest instance for B777 is solved within around 11 seconds; and the largest A380 instances are solved within 5 seconds; A320 instances within 0.4 seconds, and the B737 instances within 0.02 seconds. In comparison of the optimal objective values, the DP approaches are usually
Fig. 3. The performance distribution and runtime development plots comparing DP and MIP in VFP (a-b), and VFPW (c-d).
slightly better (less than 0.01%) than the ones obtained by MIP solvers due to the default solver optimality gap tolerance of 0.01%. Besides, computation time for DP is quite robust. As MIP solver may have performance variability in terms of computation time of up to a factor 52 times by varying parameter setting or 13 times by varying simply the random seed of the MIP solver [15], the DP approach is deterministic and parameter-free. The computation time of DP is more predictable, since the instances of similar sizes usually requires very close computation time. These properties make DP much more practical than MIP solvers for real-world applications.

In what follows, we will focus only on the more practical and computationally more challenging problem VFPW.

### 5.3 Bi-Objective Pareto Front by DP

An additional byproduct of the DP approach is that it could return all the Pareto optimal vertical profiles in terms of the two objectives of flight time and fuel consumption. This can be done by relaxing the time bound in the DP approach, i.e., to skip the time bound preprocessing. Two examples of the Pareto fronts shown in Figure 4, with A320 with 40 segments and eastwards trip, which has in total ca. 23 500 Pareto optimal solutions; and B777 with 80 segments and westwards trip, which has a total of over 85 000 Pareto optimal solutions. As can be observed from the two Pareto front plots, using a vertical profile of ca. 5.9 hours of flight time is more reasonable than the fuel-optimal vertical profile with 6.05 hours in Figure 4(a), and a vertical profile of 10.8 hours is more preferable than the fuel-optimal profile of 11 hours in Figure 4(b), since the additional fuel...
consumption is negligible. Besides, speeding up a flight with less than 5.55 hours in Figure 4(a) or a flight with less than 10.2 hours in Figure 4(b) is not recommendable since a drastic fuel increase is required for a negligible time saving. The Pareto optimal solutions illustrated in the Pareto front allows the decision maker more flexibility and practicability. Time uncertainties are common in flight industry due to runway availability, aircraft maintenance activities, and crew and passenger delays, etc. Such time uncertainties are usually recovered by optimizing only the altitude and speed assignment in the vertical planning phase instead of changing the whole flight trajectory validated by the air traffic control. In case of such a delay or a unforeseen weather disruption, the dispatcher and the pilot can simply choose from the Pareto front another best and feasible vertical profile without recomputing it again. It may require a factor of 0.5 to 5 times more computation time to finish, as shown in Figure 5. The average factor of the computation time required by the bi-objective over the single-objective DP is 3.4. However, it scales well for large instances, the computation time for the largest instance increases from 11 seconds to less than 19 seconds, since the instances that require long computation time in the single-objective optimization are probably the ones whose time bound is not so effective in pruning the solution space, thus relaxing time bound also does not bring significant time increase.

![Graph showing computation time for different DP versions](image)

**Fig. 5.** The computation time for different DP versions: the exact DP, bi-objective DP, and DP with time discretization schemes of rounding the flight time in hours to 5, 4, and 3 digits after the decimal point.

### 5.4 Further Speedup by Time Discretization

We further investigate the speed-up mechanism for our DP approach by time discretization. Three discretization levels are considered in our experiments: the
DP-5 with time discretization of 5 digits after the decimal point, i.e., 0.00001 hours (0.036 seconds) per segment; DP-4 with time discretization of 0.0001 hours (0.36 seconds) per segment; and DP-3 with time discretization of 0.001 hours (3.6 seconds) per segment. The different time discretization coarseness accelerates the DP approach differently. The finest DP-5 requires only slightly less computation time than the exact DP approach. It speeds up the exact DP by an average factor of 6.4%, and reduces the computation time for the largest instance from 11 seconds to 9 seconds, as shown in Figure 5. The DP-4 already speeds up the exact DP approach by an average factor of 3.5, and the largest instance can be solved within 1.4 seconds, i.e., almost an order of magnitude faster. The coarsest discretization DP-3 drastically accelerates the exact DP by over one order of magnitude, with an average speed-up factor of 14 times. The most drastic speed-up is usually achieved in the largest instances, e.g., the largest instance can be solved by DP-3 within 0.2 seconds with a 50 times speed-up.

Fig. 6. The percentage objective deviation from the optima computed by the exact DP approach in the four approaches: the DP-5 with time discretization of 0.00001 hours (0.036 seconds), DP-4 with time discretization of 0.0001 hours (0.36 seconds), DP-3 with time discretization of 0.001 hours (3.6 seconds), and the MIP solver with a cutoff time of four hours.

The percentage objective deviation between the optimal solutions found by the different time discretization schemes and the exact DP approach is illustrated in Figure 6. The best solutions found by the MIP solver with a cutoff time of 4 hours are also shown as a reference. It is noteworthy that the optimal solution found by the MIP solver is usually higher than the optimum found by the exact DP approach due to a default optimality tolerance of 0.01%, i.e., the solution is considered optimal when the MIP gap is within 0.01%. As shown in Figure 6, indeed 90% of the instances are solved by the MIP solver within 0.01% from the exact optimum, while there are still outliers which are not solved within 4 hours, and the maximum deviation from optimality is around 2%. The optimal objective deviation of the three time discretization schemes is higher
when the discretization is coarser. The objective deviation of DP-5 is ignorable, since there are only 7 out of 72 instances whose computed solution differs from the exact optimum, and only 4 instances with a deviation higher than 0.0001%, with maximum deviation 0.0007%. The DP-4 also found the exact optimal solution in more than half of benchmark instances, and its objective deviation is still better than the MIP solver. Its maximum deviation is 0.016%. Even coarsest discretization scheme DP-3 has a reasonable deviation with a median of 0.02% from the exact optimum, and its maximum deviation is 0.34%. In computation time critical cases, the time discretization scheme is a viable approach to speed up our proposed DP approach.

6 Conclusions

In this work, we proposed a dynamic programming (DP) approach for the vertical flight planning problem, which concerns assigning optimal altitude and speed to each composing segment of a flight trajectory. The DP approach is deterministic, parameter-free, and can solve real-world vertical flight planning problems with realistic sizes to exact optimality within a reasonable computation time. The largest benchmark instances are solved within around 11 seconds, where it takes the state-of-the-art MIP solver 4 hours with a 2% gap. We further explored the use of the DP approach to compute the bi-objective Pareto front in terms of fuel consumption and flight time, and DP can achieve this task with a reasonable computation time of up to 19 seconds for the largest benchmark instances. We also analyzed the speed-up technique for the DP by a time discretization approach. The time discretization can significantly speed up the exact DP approach by a factor up to 50 without severe optimal objective deviation. Our experimental results show that the proposed dynamic programming approach is promising and practically viable for the flight planning problems.

Acknowledgments Zhi Yuan acknowledges partial support by BMBF Verbundprojekt E-Motion. The authors thank Anton Kaier and Swen Schlobach from Lufthansa Systems AG for providing the aircraft performance data and upper air data, and their insights on the flight planning praxis.

References


