Submitted to *Transportation Science*
manuscript (Please, provide the manuscript number!)

Branch-and-price for drone delivery service planning in urban airspace

Michael W. Levin
Department of Civil, Environmental, and Geo- Engineering, University of Minnesota, mlevin@umn.edu
David Rey
SKEMA Business School, Université Côte d'Azur, Sophia Antipolis Campus, France, david.rey@skema.edu

Unmanned aerial vehicles or drones are increasing in use both for commercial and casual purposes. Although drone traffic management is mostly absent, as drone use increases aerial conflicts are likely to also increase. This paper studies the problem of planning drone delivery service through an urban air traffic network space. The urban air traffic network is assumed to mostly be the airspace above existing roads, and is modeled as a transportation network with multiple flight levels. Drone flights are modeled as individual trip requests with origins, destinations, and time windows. We present a novel integer linear programming formulation for this drone delivery service planning problem. The main contribution of this paper is developing a branch-and-price algorithm to solve the formulation, as the number of decision variables grows quickly with the problem size. We investigate three variations of branch-and-price, including branching on trajectory assignment variables, branching rules related to served requests, and a primal heuristic to quickly find integer feasible solutions. Numerical results show the limits of the integer linear programming formulation and the benefits of the primal heuristic in finding a good feasible solution.

Key words: Unmanned aerial vehicles, airspace management, drone delivery, service planning, branch-and-price
History:

1. Introduction

Unmanned aerial vehicles (UAVs) or drones have been available both commercially and to hobbyists for many years now. UAVs have been studied for potential use in diverse applications including roadway monitoring (Coifman et al. 2006) and freight transportation (Klochkov and Karpov 2018). The potential for UAVs to carry goods efficiently has actually been studied extensively for viability (Aurambout, Gkoumas, and Ciuffo 2019). Several studies also predicted reductions in environmental impacts from replacing trucks with UAVs (Goodchild and Toy 2018, Figliozzi 2017,

Kirschstein 2020). As UAV applications increase and technology improves, the number of UAV flights is also likely to increase (Sándor 2019) along with the emergence of Unmanned aircraft systems Traffic Management (UTM) (NASA 2021). A large number of UAV flights increases the potential for midair collisions caused by UAVs (Zhang et al. 2018a). Indeed, developing collision avoidance systems for UAVs has been extensively studied (Huang, Teo, and Tan 2019). Midair collisions between commercial aircraft are typically avoided through strict separation requirements between enroute aircraft, and these requirements have led to research on minimizing aircraft conflicts (e.g Rey and Hijazi 2017). Typical controls used to avoid conflicts include changing the aircraft's altitude, speed, or lateral path (Dias, Hijazi, and Rey 2022). However, reasonable separation requirements for UAVs (Weibel, Edwards, and Fernandes 2011), and corresponding methods of ensuring separation (Jenie et al. 2016), are still being established.

UTM is still in its early years. In urban settings, buildings, power lines, and other obstructions occupy the airspace, and depending on the location, may exceed several hundred feet above ground level. Due to the power required for climbing, especially for drones carrying goods (Klochkov and Karpov 2018), drones may not want to climb above urban buildings. However, the airspace above roads is typically less obstructed and free for drones. Therefore, it is intuitive to expect that urban airspace will emerge above road networks and can thus be represented as a network of nodes and links. Nodes may represent the airspace above traffic intersections and links may represent the airspace above roads connecting those nodes. Unlike standard road networks, multiple flight levels, or altitudes of travel, may exist for drones.

The purpose of this study is to address a drone delivery service planning problem in the context of UTM. We consider an urban air traffic network with commodities representing goods to be carried via UAVs. We take the perspective of a UTM network manager which aims to balance service throughput and operations costs. We further assume that after takeoff and before landing, drones fly horizontally at an assigned flight level, similar to ATM operations in civil aviation. Links of the UTM network are assumed to have known capacities and, once airborne, drones are assumed to travel at a fixed speed along their trajectory. Hence, to prevent airborne congestion which may cause travel speed reductions, drones must be assigned to conflict-free space-time trajectories in the urban air traffic network. In this context, given a set of drone flight requests, each with a given origin and destination, departure and arrival time windows, the problem is to assign optimal trajectories for all requests served subject to network design constraints. We next review the literature and present the background of this study before outlining our contributions.

1.1. State-of-the-art

Drones have many potential applications, and these applications have received some attention in the literature (Otto et al. 2018). Most previous work on UAVs can largely be categorized into two

groups: vehicle routing problems with drones and network design for UAV logistics. In practical use, the coordinated control of a large number of drones is likely to occur within the context of vehicle routing problems (Toth and Vigo 2001), in which a commercial drone operator needs to visit a number of locations for freight delivery or other purposes. Unlike most road vehicles, drones have more restrictive constraints in terms of flying time before refueling and payload weight (which also significantly affects fuel consumption). These restrictions affect vehicle routing considerably. For instance, Murray and Chu (2015) proposed that drones operate from mobile trucks, and proposed a combined truck and drone traveling salesman problem. This combined approach was later studied by others as well, both for a single drone (Carlsson and Song 2018, Ha et al. 2018, Agatz, Bouman, and Schmidt 2018) and multiple UAVs (Wang, Poikonen, and Golden 2017, Boysen, Schwerdfeger, and Weidinger 2018). The use of trucks as a mobile depot for the drones creates a particularly interesting problem structure (Bouman, Agatz, and Schmidt 2018). The problem can be formulated as an integer program, and branch-and-price techniques have been applied to it (Wang and Sheu 2019). Other studies (Dorling et al. 2016, Zhang et al. 2018b) focused on drone routing from fixed depots, including awareness of variable weather conditions (Radzki, Thibbotuwawa, and Bocewicz 2019). There are many studies already on different variations of UAV routing as well as different solution approaches. These are mostly orthogonal to the network optimization problem considered in this study. Specifically, we focus on the management of an urban air traffic network where flight request for UAVs are to be served by a network manager. Hence, we assume that UAV trips have already been determined and instead focus our attention on network management and drone delivery service planning.

Network optimization for UTM is an emerging field of research which builds on classical air traffic management (ATM) and air traffic control (ATC) and intersects with the broader literature on network flow and design problems (Otto et al. 2018). Kopardekar et al. (2016) discussed different options for the design and control of UAV airspace conceptually, but did not work on optimal use. Mohamed Salleh and Low (2017) extended the concepts to consider individual blocks of airspace for easier risk modeling and discussion of origin and destination points for buildings. Mohamed Salleh et al. (2018) then discussed different levels of control, and evaluated the tradeoff between more control and less control through simulation. Ren et al. (2017) created a simulation of UTM, and Peinecke and Kuenz (2017) developed a simulation of UAV trips in realistic airspace and proposed a method of modifying trajectories to reduce air conflicts. Rios et al. (2016) conducted live tests of collision avoidance between UAVs operating in close proximity.

Conflict detection and resolution has also been studied using optimization techniques. As in ATM systems for civil aviation, conflicts between UAVs extend beyond avoiding actual collisions to maintaining separation minima between any two individual UAVs (D'Souza et al. 2016). Although

individual UAVs are small, separation minima can result in active capacity constraints (Cho and Yoon 2018). Conflict resolution can occur both in the short-term and the pre-departure (flight planning) stage. Ong and Kochenderfer (2017) used a Markov decision process for short-term conflict resolution, and Ho et al. (2018) developed a tabu search heuristic for conflict detection and resolution. Alharbi et al. (2020) proposed rules for short-term conflict resolution (e.g. hovering) as well as conflict resolution at the flight planning stage. Other studies focused on long-term conflict resolution. Alejo et al. (2013) studied the trajectory optimization problem under the assumption that UAVs follow a set of waypoints and used particle swarm optimization as a solution method. Vera et al. (2016) also studied this problem and applied a rolling horizon approach to solve it. Kuru et al. (2019) compared different delivery methods and different routing policies for UAVs.

The work of Chin et al. (2020) and Chin et al. (2021) is similar to our study. They also develop optimization formulations for assigning trajectories to drones in an air traffic network. The main differences between this paper and their studies is our focus on an urban airspace, such as the airspace above an urban road network. The formulation is based on a network structure for the airspace, with specific links or airways of travel. It is not necessary that these links exist over urban roads; they could be arbitrarily chosen. We note that prior to area navigation (RNAV), the conventional air traffic system relied on airways formed by VHF omnidirectional radio (VOR) beacons, which also formed a network structure. This network structure creates intersections representing air conflict points where UAV traffic must be coordinated to avoid collisions. Specifically, in this study, we address the problem of assigning UAV trip requests to "reserved" conflict-free, space-time trajectories (Ho et al. 2019). We assume that we are given a set of UAV trip requests specifying their origin, destination, and departure and arrival time windows. The goal is to maximize the number of requests served while balancing operations costs within a capacity-limited airspace. Hence, our focus is on the coordination of multiple UAV trips, not the detailed control of a single drone (Chamseddine et al. 2012). We propose mixed-integer programming formulations for this UTM network optimization problem along with a customized branch-and-price algorithm for improving computational scalability. Without it, the number of variables in such formulations increases with the number of trips and the size of the airspace, resulting in the optimization problem being computationally intractable for large-scale but realistic urban networks.

1.2. Contributions

This study makes the following contributions to the field. First, we formulate a drone delivery service planning problem as an integer linear program which main binary variables indicate whether each UAV flight request uses a specific link at a specific time. These variables define a UAV trajectory, and conflict avoidance and capacity constraints are incorporated. This formulation

appears to be novel to the literature, but the number of variables increases with the number of requests, the number of flight levels, and the network size. Therefore, our main contribution is in developing a novel branch-and-price algorithm to solve this drone delivery service planning problem. We reformulate the problem in terms of trajectory assignments, and use a column generation approach to generate trajectories that might improve the objective value. Branching is used to ensure that an integer solution is reached. We also present several heuristics that exploit the problem structure, and numerical results suggest that these heuristics and the branch-and-price algorithm are effective at finding competitive solutions.

The remainder of this paper is organized as follows. Section 2 introduces the network structure and presents a simple integer linear programming formulation of the trajectory optimization problem. Section 3 develops a branch-and-price algorithm that is suitable for larger networks or with a larger number of requests. The standard branch-and-price approach is supplemented with several heuristics exploiting the problem structure. Section 4 evaluates the performance of the proposed algorithm and heuristics, both against each other and with solving the integer linear program directly using a commercial solver. We present conclusions in Section 5.

2. Problem formulation

Consider a network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$. All demand departs and arrives at zones, and let $\mathcal{Z} \subseteq \mathcal{N}$ be the set of zones. Let Γ_n^- and Γ_n^+ denote the sets of incoming and outgoing links for node n, respectively. To simplify the notation, we also overload Γ_i^- and Γ_i^+ to be the sets of incoming and outgoing links for the downstream node of link i, respectively. Each link $i \in \mathcal{A}$ has length ℓ_i and capacity C_i. Links can be traversed at one of multiple flight levels. Let $\mathcal{F}$ be the set of flight levels. Each flight level f has constant speed v_f. Each flight level is assumed to have the same speed to simplify separation requirements on links, but different flight levels may have different speeds. Therefore, link travel times at a given flight level are constant. We discretize time, and let $\{0, \ldots, T\}$ be the time horizon.

We assume that on departure, drones will immediately climb to their assigned flight level and remain there until reaching their destination. This assumption is easily valid for rotary-wing drones, which can climb vertically. Drones are restricted in the set of permissible flight levels based on their maximum altitude and range of speeds. All links are directed. Let $\mathcal{F}_i$ be the set of flight levels which are permissible on link i. $\mathcal{F}_i \subsetneq \mathcal{F}$ is possible due to obstacles to navigation.

Let $\mathcal{R}$ be the set of requests. Requests are modeled individually due to differences in parameters, which are used to characterize each request. Request r has origin $o_r \in \mathcal{Z}$ and destination $d_r \in \mathcal{Z}$. Let e_r be the earliest departure time of r. The arrival time window of r is specified by $[l_r, u_r]$. The set of valid flight levels for r is $\mathcal{F}_r$. It is possible to have $\mathcal{F}_r \subsetneq \mathcal{F}$ due to airspace limitations on altitude or speed.

2.1. Link flow constraints

Let $x^{\uparrow}_{if}(t)$ and $x^{\downarrow}_{if}(t)$ be the number of drones crossing the upstream and downstream ends of link i and flight level f, respectively. These flows are restricted by link capacity:

$$x^{\uparrow}_{if}(t) \leq C_i \qquad \forall i \in \mathcal{A}, \forall f \in \mathcal{F}_i, \forall t \in \{0, \ldots, T\} \tag{1}$$

$$x^{\downarrow}_{if}(t) \leq C_i \qquad \forall i \in \mathcal{A}, \forall f \in \mathcal{F}_i, \forall t \in \{0, \ldots, T\} \tag{2}$$

with the constraint that $x^{\uparrow}_{if}(t) = 0$ if $f \notin \mathcal{F}_i$. $x^{\uparrow}_{if}(t)$ and $x^{\downarrow}_{if}(t)$ are related to each other, separated by the link travel time:

$$x^{\downarrow}_{if}\left(t + \frac{\ell_i}{v_f}\right) = x^{\uparrow}_{if}(t) \qquad \forall i \in \mathcal{A}, \forall f \in \mathcal{F}_i, \forall t \in \{0, \ldots, T\} \tag{3}$$

The conservation of flow constraint is

$$\sum_{i \in \Gamma^-_n} x^{\downarrow}_{if}(t) = \sum_{j \in \Gamma^+_n} x^{\uparrow}_{jf}(t) \qquad \forall n \in \mathcal{N}, \forall f \in \mathcal{F}, \forall t \in \{0, \ldots, T\} \tag{4}$$

We require that $x^{\uparrow}_{if}(t) \in \mathbb{Z}_+$ and $x^{\downarrow}_{if}(t) \in \mathbb{Z}_+$. These constraints will be made clear through later development.

2.2. Node conflict constraints

Let $y_{ijf}(t)$ be the number of drones traveling from the downstream end of link i to the upstream end of link j at time t at flight level f. $y_{ijf}(t)$ is related to $x^{\uparrow}_{jf}(t)$ and $x^{\downarrow}_{if}(t)$ via

$$x^{\uparrow}_{if}(t) = \sum_{j \in \Gamma^+_i} y_{ijf}(t) \qquad \forall i \in \mathcal{A}, \forall f \in \mathcal{F}_i \cup \mathcal{F}_j, \forall t \in \{0, \ldots, T\} \tag{5}$$

$$x^{\downarrow}_{jf}(t) = \sum_{i \in \Gamma^-_j} y_{ijf}(t) \qquad \forall j \in \mathcal{A}, \forall f \in \mathcal{F}_i \cup \mathcal{F}_j, \forall t \in \{0, \ldots, T\} \tag{6}$$

The strict equalities in equations (3)–(6) indicate that drones must travel exactly at the required speed v_f. They cannot wait at nodes until conflicting traffic has passed. Instead, their trajectories must be planned for continuous forward motion. Although rotary-wing UAVs could hover to increase the number of feasible trajectories, fixed-wing UAVs would not be able to. Furthermore, varying link travel speeds creates the possibility of congestion on links. Although such congestion might be modeled with the link transmission model (Yperman, Logghe, and Immers 2005), its additional complexity when solving this problem is left for future work.

A turning movement is a movement from i to j. A turning movement occurs from i to j at flight level f at time t if $y_{ijf}(t) > 0$. To prevent congestion and collisions at nodes, we do not permit conflicting turning movements to use the same flight level at the same time. Let $\phi_{ijf}(t) \in \{0, 1\}$

indicate whether movement from (i,j) at flight level f is permitted at time t. $\phi_{ijf}(t)$ can be related to $y_{ijf}(t)$ via

$$\phi_{ijf}(t) \geq \frac{1}{|\mathcal{R}|} y_{ijf}(t) \qquad \forall (i,j) \in \mathcal{A}^2, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{7}$$

Let $\mathcal{C}_{ij}$ be the set of turning movements that conflict with (i,j). Then conflicts can be prevented by requiring

$$\phi_{ijf}(t) + \sum_{(i',j') \in \mathcal{C}_{ij}} \phi_{i'j'f}(t) \leq 1 \qquad \forall (i,j) \in \mathcal{A}^2, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{8}$$

2.3. Request-based link flows

For an initial naïve formulation of the trajectory optimization problem, we track the flows of individual requests through each link of the network. Let $\chi_{if}^{r\uparrow}(t) \in \{0,1\}$ and $\chi_{if}^{r\downarrow}(t) \in \{0,1\}$ indicate whether request r enters or exits link i at flight level f at time t. This approach will lead to an exact method for optimizing trajectories, but the number of variables will grow quite large. The number of $\chi_{if}^{r\uparrow}(t)$ variables is $O(|\mathcal{R}| \times |\mathcal{A}| \times |\mathcal{F}| \times T)$. Then $x_{if}^{\uparrow}(t)$ and $x_{if}^{\downarrow}(t)$ can be written in terms of $\chi_{if}^{r\uparrow}(t)$ and $\chi_{if}^{r\downarrow}(t)$:

$$x_{if}^{\uparrow}(t) = \sum_{r \in \mathcal{R}} \chi_{if}^{r\uparrow}(t) \qquad \forall i \in \mathcal{A}, \forall f \in \mathcal{F}_i, \forall t \in \{0,\dots,T\} \tag{9}$$

$$x_{if}^{\downarrow}(t) = \sum_{r \in \mathcal{R}} \chi_{if}^{r\downarrow}(t) \qquad \forall i \in \mathcal{A}, \forall f \in \mathcal{F}_i, \forall t \in \{0,\dots,T\} \tag{10}$$

with

$$\chi_{if}^{r\downarrow}\left(t + \frac{\ell_i}{v_f}\right) = \chi_{if}^{r\uparrow}(t) \qquad \forall i \in \mathcal{A}, \forall f \in \mathcal{F}_i, \forall t \in \{0,\dots,T\} \tag{11}$$

Let $\gamma_{ijf}^{r}(t) \in \{0,1\}$ indicate whether drone r travels from link i to link j at time t at flight level f. Then $\gamma_{ijf}^{r}(t)$ can be written in terms of $\chi_{if}^{r\uparrow}(t)$ and $\chi_{if}^{r\downarrow}(t)$:

$$\gamma_{ijf}^{r}(t) \leq \chi_{if}^{r\downarrow}(t) \qquad \forall r \in \mathcal{R}, \forall i,j \in \mathcal{A}, \forall f \in \mathcal{F}_i \cup \mathcal{F}_j, \forall t \in \{0,\dots,T\} \tag{12}$$

$$\gamma_{ijf}^{r}(t) \leq \chi_{jf}^{r\uparrow}(t) \qquad \forall r \in \mathcal{R}, \forall i,j \in \mathcal{A}, \forall f \in \mathcal{F}_i \cup \mathcal{F}_j, \forall t \in \{0,\dots,T\} \tag{13}$$

$$\gamma_{ijf}^{r}(t) \geq \chi_{if}^{r\downarrow}(t) + \chi_{if}^{r\uparrow}(t) - 1 \qquad \forall r \in \mathcal{R}, \forall i,j \in \mathcal{A}, \forall f \in \mathcal{F}_i \cup \mathcal{F}_j, \forall t \in \{0,\dots,T\} \tag{14}$$

Constraints (12)–(14) ensure that $\gamma_{ijf}^{r}(t) = 1$ if $\chi_{if}^{r\downarrow}(t) = \chi_{jf}^{r\uparrow}(t) = 1$. $y_{ij}(t)$ can be written in terms of $\gamma_{ijf}^{r}(t)$ as follows:

$$y_{ijf}(t) = \sum_{r \in \mathcal{R}} \gamma_{ijf}^{r}(t) \qquad \forall i,j \in \mathcal{A}, \forall f \in \mathcal{F}_i \cup \mathcal{F}_j, \forall t \in \{0,\dots,T\} \tag{15}$$

2.4. Departure tracking

A drone departs when it enters one of the links outgoing from its origin node o_r, denoted by the set $\Gamma^+_{o_r} \subseteq \mathcal{A}$. Each request has an earliest departure time e_r. That is modeled by

$$\chi^{r\uparrow}_{if}(t) = 0 \qquad \forall t < e_r, \forall i \in \Gamma^+_{o_r}, \forall f \in \mathcal{F}_r \tag{16}$$

When implementing, the above constraint can be simplified by removing $\chi^{r\uparrow}_{if}(t)$ variables from the formulation when $t < e_r$. A drone can only depart once. After departure, the drone is assumed to immediately climb vertically to its initial flight level. Let $z_r \in \{0,1\}$ be a binary variable representing if request $r \in \mathcal{R}$ is served or not. Observe that $z_r = \sum_{i \in \Gamma^+_{o_r}} \sum_{f \in \mathcal{F}_r} \sum_{t=e_r}^{T} \chi^{r\uparrow}_{if}(t)$. For each request $r \in \mathcal{R}$, we require only: $z_r \leq 1$.

Drones arrive once they reach the downstream end of a link connected to their destination node d_r. A drone that departs must arrive within its arrival time window. The expression $\sum_{i \in \Gamma^-_{d_r}} \sum_{f \in \mathcal{F}_r} \sum_{t=l_r}^{u_r} \chi^{r\downarrow}_{if}(t)$ indicates whether request r arrives. Therefore, the arrival and departure of request r can be constrained as

$$\sum_{i \in \Gamma^-_{d_r}} \sum_{f \in \mathcal{F}_r} \sum_{t=l_r}^{u_r} \chi^{r\downarrow}_{if}(t) = \sum_{i \in \Gamma^+_{o_r}} \sum_{f} \sum_{t=e_r}^{T} \chi^{r\uparrow}_{if}(t) \qquad \forall r \in \mathcal{R} \tag{17}$$

The arrival / departure of trip request $r \in \mathcal{R}$ is also used to define z_r:

$$\sum_{i \in \Gamma^-_{d_r}} \sum_{f \in \mathcal{F}_r} \sum_{t=l_r}^{u_r} \chi^{r\downarrow}_{if}(t) = z_r \qquad \forall r \in \mathcal{R} \tag{18}$$

The requirement that $z_r \in \{0,1\}$ ensures that each drone departs at most once. Variable z_r is 0 if the trip request $r \in \mathcal{R}$ is not served for optimality or feasibility reasons.

2.5. Objective function

There are several potential objectives for optimizing drone delivery service planning in the UTM network. The simplest objective is to maximize the number of served requests. When demand exceeds network capacity, this objective is interesting and valuable to solve. However, when all demand can be served, finding an assignment that serves all demand may be easy. We will describe a heuristic in Section 3 and demonstrate its effectiveness in Section 4. In that case, we want to find an assignment that minimizes the travel cost. Since most of the travel cost is related to flight time, we focus on reducing the time spent traveling (which does not include time spent waiting to depart at the origin). To incorporate both objectives into one formulation, we define a parameter α which specifies the degree to which we prioritize serving all requests. The travel times are then weighted by $(1-\alpha)$. The time that request r spends traveling can be obtained by $\sum_{i \in \mathcal{A}} \sum_{f \in \mathcal{F}_i} \sum_{t=e_r}^{T} \chi^{r\uparrow}_{if}(t) \left(\frac{\ell_i}{v_f}\right)$. The link travel time $\frac{\ell_i}{v_f}$ is included only when $\chi^{r\uparrow}_{if}(t) = 1$.

Furthermore, we define p_r to be the profit associated with request r. If we want to only maximize the number of requests served, then we can set $p_r = 1$ for all r. However, in freight operations it is common for customers to be given the option to pay extra for higher priority for fast delivery. In that case, we could set the profit for some requests to be higher to indicate their greater priority. Therefore, the objective function is

$$\max Z = \alpha \sum_{r\in\mathcal{R}} p_r z_r - (1-\alpha) \sum_{r\in\mathcal{R}} \sum_{i\in\mathcal{A}} \sum_{f\in\mathcal{F}_i} \sum_{t=e_r}^{T} \chi_{if}^{r\uparrow}(t) \left(\frac{\ell_i}{v_f}\right) \tag{19}$$

2.6. Link-based formulation

We now present a link-based formulation for the drone delivery service planning problem. We have made some simplifications to the equations developed above: The link flow variables $x_{if}^{\uparrow}(t)$ and $x_{if}^{\downarrow}(t)$ are redundant when $\chi_{if}^{r\uparrow}(t)$ and $\chi_{if}^{r\downarrow}(t)$ are included. The same can be said of $y_{ijf}(t)$ and $\gamma_{ijf}^{r}(t)$. We have therefore omitted $x_{if}^{\uparrow}(t)$m $x_{if}^{\downarrow}(t)$, and $y_{ijf}(t)$ from the formulation to reduce the number of variables.

$$\max \quad Z = \alpha \sum_{r\in\mathcal{R}} p_r z_r - (1-\alpha) \sum_{r\in\mathcal{R}} \sum_{i\in\mathcal{A}} \sum_{f\in\mathcal{F}_i} \sum_{t=e_r}^{T} \chi_{if}^{r\uparrow}(t) \left(\frac{\ell_i}{v_f}\right) \tag{20a}$$

$$\text{s.t.} \quad \sum_{i\in\Gamma_{o_r}^{+}} \sum_{f\in\mathcal{F}_r} \sum_{t=e_r}^{T} \chi_{if}^{r\uparrow}(t) = z_r \qquad \forall r \in \mathcal{R} \tag{20b}$$

$$\sum_{i\in\Gamma_{d_r}^{-}} \sum_{f\in\mathcal{F}_r} \sum_{t=l_r}^{u_r} \chi_{if}^{r\downarrow}(t) = \sum_{i\in\Gamma_{o_r}^{+}} \sum_{f\in\mathcal{F}_r} \sum_{t=e_r}^{T} \chi_{if}^{r\uparrow}(t) \quad \forall r \in \mathcal{R} \tag{20c}$$

$$\chi_{if}^{r\downarrow}\left(t + \frac{\ell_i}{v_f}\right) = \chi_{if}^{r\uparrow}(t) \qquad \forall r \in \mathcal{R}, \forall i \in \mathcal{A}, \forall f \in \mathcal{F}_i, \forall t \in \{0,\dots,T\} \tag{20d}$$

$$\sum_{r\in\mathcal{R}} \chi_{if}^{r\uparrow}(t) \le C_{if} \qquad \forall i \in \mathcal{A}, \forall f \in \mathcal{F}_i, \forall t \in \{0,\dots,T\} \tag{20e}$$

$$\sum_{r\in\mathcal{R}} \chi_{if}^{r\downarrow}(t) \le C_{if} \qquad \forall i \in \mathcal{A}, \forall f \in \mathcal{F}_i, \forall t \in \{0,\dots,T\} \tag{20f}$$

$$\sum_{i\in\Gamma_{n}^{-}} \chi_{if}^{r\downarrow}(t) = \sum_{j\in\Gamma_{n}^{+}} \chi_{if}^{r\uparrow}(t) \qquad \forall n \in \mathcal{N}, \forall r \in \mathcal{R}, \forall f \in \mathcal{F}_r, \forall t \in \{0,\dots,T\} \tag{20g}$$

$$\gamma_{ijf}^{r}(t) \le \chi_{if}^{r\downarrow}(t) \qquad \forall r \in \mathcal{R}, \forall (i,j) \in \mathcal{A}^2, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{20h}$$

$$\gamma_{ijf}^{r}(t) \le \chi_{jf}^{r\uparrow}(t) \qquad \forall r \in \mathcal{R}, \forall (i,j) \in \mathcal{A}^2, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{20i}$$

$$\gamma_{ijf}^{r}(t) \ge \chi_{if}^{r\downarrow}(t) + \chi_{if}^{r\uparrow}(t) - 1 \qquad \forall r \in \mathcal{R}, \forall (i,j) \in \mathcal{A}^2, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{20j}$$

$$\phi_{ijf}(t) \ge \frac{1}{|\mathcal{R}|} \sum_{r\in\mathcal{R}} \gamma_{ijf}^{r}(t) \qquad \forall (i,j) \in \mathcal{A}^2, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{20k}$$

$$\phi_{ijf}(t) + \sum_{(i',j') \in \mathcal{C}_{ij}} \phi_{i'j'f}(t) \leq 1 \qquad \forall (i,j) \in \mathcal{A}^2, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{20l}$$

$$\gamma^r_{ijf}(t) \in \{0,1\} \qquad \forall r \in \mathcal{R}, \forall (i,j) \in \mathcal{A}^2, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{20m}$$

$$\phi_{ijf}(t) \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}^2, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{20n}$$

$$\chi^{r\uparrow}_{if}(t), \chi^{r\downarrow}_{if}(t) \in \{0,1\} \qquad \forall r \in \mathcal{R}, \forall i \in \mathcal{A}, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{20o}$$

$$z_r \in \{0,1\} \qquad \forall r \in \mathcal{R} \tag{20p}$$

Although formulation (20) is fairly straightforward, it appears to be novel to the literature. Formally, the number of variables and constraints in the link-based formulation is $O(|\mathcal{A}| \times |\mathcal{F}| \times |\mathcal{R}| \times T)$. However, formulation (20) is limited by how the number of variables increases linearly with the number of links and requests. The main contribution of this paper is in developing a more scalable algorithm to solve problem (20).

3. Branch-and-price algorithm

To solve problem (20) on large networks, we develop a branch-and-price (BP) algorithm. First, we reformulate problem (20) in terms of paths. Although this increases the number of variables, we will show that the linear relaxation of this path-based formulation can be solved efficiently by column generation (CG). We then develop customized branching rules and primal heuristics to find optimal or competitive integer solutions.

3.1. Path-based formulation

Let π be a trajectory in the network which includes an ordered set of connected links and a departure time. Recall that because link travel times are fixed, the departure time is sufficient information to determine the enter and exit times on all subsequent links. Let Π_r be the set of feasible paths for request r. Feasibility includes paths that start at o_r and end at d_r, depart on or after e_r, and arrive in the interval $[l_r, u_r]$. Hence, a path $\pi \in \Pi_r$ corresponds to both a valid departure time and a sequence of links from the origin to the destination of request $r \in \mathcal{R}$. Let $\mathcal{F}_\pi$ be the set of valid flight levels for path π. F_π can be determined via

$$\mathcal{F}_\pi = \bigcap_{i \in \pi} \mathcal{F}_i \tag{21}$$

Let $h^r_{\pi f} \in \{0,1\}$ indicate whether request r is assigned to path π at flight level f. Each request can only be assigned to at most one path and one flight level:

$$\sum_{\pi \in \Pi_r} \sum_{f \in \mathcal{F}_\pi} h^r_{\pi f} \leq 1 \qquad \forall r \in \mathcal{R} \tag{22}$$

It is possible that a request is not assigned to a path at all, and thus would not be served. Because π specifies the arrival times on each link, it can be determined whether π uses link i or turning movement (i,j) at time t. Let $\delta_i^\pi(t) \in \{0,1\}$ indicate whether π enters link i at time t. Similarly, let $\delta_{ij}^\pi(t) \in \{0,1\}$ indicate whether π moves from link i to link j at time t. Then constraint (20d) can be rewritten as

$$\sum_{r\in\mathcal{R}}\sum_{\pi\in\Pi_r} \delta_i^\pi(t) h_{\pi f}^r \le C_{if} \qquad \forall i \in \mathcal{A}, \forall f \in \mathcal{F}_i, \forall t \in \{0,\dots,T\} \tag{23}$$

Because link travel times are constant at each flight level, constraint (20e) is equivalent to constraint (20d). Constraint (20j) can be replaced by

$$\phi_{ijf}(t) \ge \sum_{\pi\in\Pi_r} \delta_{ij}^\pi(t) h_{\pi f}^r \qquad \forall r \in \mathcal{R}, \forall (i,j) \in \mathcal{A}^2, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{24}$$

Constraint (20k) still applies. This results in the following path-based formulation:

$$\max \quad Z = \sum_{r\in\mathcal{R}}\sum_{\pi\in\Pi_r}\sum_{f\in\mathcal{F}_r} \left[h_{\pi f}^r \left(\alpha p_r - (1-\alpha)\tau_{\pi f}\right)\right] \tag{25a}$$

$$\text{s.t.} \quad \sum_{\pi\in\Pi_r}\sum_{f\in\mathcal{F}_r} h_{\pi f}^r = z_r \qquad \forall r \in \mathcal{R} \tag{25b}$$

$$\sum_{r\in\mathcal{R}}\sum_{\pi\in\Pi_r} \delta_i^\pi(t) h_{\pi f}^r \le C_{if} \qquad \forall i \in \mathcal{A}, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{25c}$$

$$\phi_{ijf}(t) \ge \sum_{\pi\in\Pi_r} \delta_{ij}^\pi(t) h_{\pi f}^r \qquad \forall r \in \mathcal{R}, \forall (i,j) \in \mathcal{A}^2, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{25d}$$

$$\phi_{ijf}(t) + \sum_{(i',j')\in\mathcal{C}_{ij}} \phi_{i'j'f}(t) \le 1 \qquad \forall (i,j) \in \mathcal{A}^2, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{25e}$$

$$\phi_{ijf}(t) \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A}^2, \forall f \in \mathcal{F}, \forall t \in \{0,\dots,T\} \tag{25f}$$

$$h_{\pi f}^r \in \{0,1\} \qquad \forall r \in \mathcal{R}, \forall \pi \in \Pi_r, \forall f \in \mathcal{F}_\pi \tag{25g}$$

$$z_r \in \{0,1\} \qquad \forall r \in \mathcal{R} \tag{25h}$$

where $\tau_{\pi f}$ is the travel time of path π at flight level f so that objective (25a) is equal to objective (19). It is easy to find a trivial feasible solution — set $h_{\pi f}^r = 0$ for all r, π, and f. Finding the optimal solution is more difficult. The set Π_r could potentially be very large for each request r. We next present the BP algorithm to solve problem (25).

3.2. Restricted master problem

Let $\bar{\Pi}_r$ be the set of restricted paths for trip request $r \in \mathcal{R}$. Consider the linear programming relaxation of (25) obtained by relaxing constraints (25f)–(25h):

$$\max \quad Z = \sum_{r\in\mathcal{R}}\sum_{\pi\in\bar{\Pi}_r}\sum_{f\in\mathcal{F}_r} \left[h_{\pi f}^r \left(\alpha p_r - (1-\alpha)\tau_{\pi}\right)\right] \tag{26a}$$

$$\text{s.t.} \quad 0 \leq \phi_{ijf}(t) \leq 1 \qquad \forall (i,j) \in \mathcal{A}^2, \forall f \in \mathcal{F}, \forall t \in \{0,\ldots,T\} \qquad (26b)$$

$$0 \leq h^r_{\pi f} \leq 1 \qquad \forall r \in \mathcal{R}, \forall \pi \in \bar{\Pi}_r, \forall f \in \mathcal{F}_\pi \qquad (26c)$$

$$0 \leq z_r \leq 1 \qquad \forall r \in \mathcal{R} \qquad (26d)$$

(25b)–(25e)

Formulation (26) is the restricted master problem (RMP) of the CG procedure. If the size of $\bar{\Pi}_r$ is kept bounded, the number of variables is $O(|\mathcal{R}| \times |\mathcal{F}|)$ and the number of constraints is $O(|\mathcal{A}| \times T)$. We next give the following result which will reduce the level of variable branching in the BP algorithm.

PROPOSITION 1. *Suppose that a feasible solution of Formulation* (26) *is such that* $[h^r_{\pi f}]$ *for all* $r \in \mathcal{R}$, $\pi \in \Pi_r$, *and* $f \in \mathcal{F}_\pi$ *is integer. Then, Formulation* (26) *admits an integer feasible solution.*

Proof. First observe that if $[h^r_{\pi f}]$ is integer, then $h^r_{\pi f} \in \{0,1\}$ because of constraint (26c); and $[z_r]$ is also binary due to constraint (25a). Assume that $[\phi_{ijf}(t)]$ for all $(i,j) \in \mathcal{A}^2$, $f \in \mathcal{F}$, and $t \in \{0,\ldots,T\}$ is fractional (otherwise the proposition is trivial). Consider the case where $\delta^\pi_{ij}(t) = 1$, because if $\delta^\pi_{ij}(t) = 0$ then $h^r_{\pi f}$ has no required relationship with $\phi_{ijf}(t)$. If $h^r_{\pi f} = 1$, then $\phi_{ijf}(t) = 1$ is required by constraint (25d). If $h^r_{\pi f} = 0$, then $\phi_{ijf}(t) > 0$ may be feasible, but $\phi_{ijf}(t) = 0$ is also feasible by constraint (25d). Therefore if $h^r_{\pi f} \in \{0,1\}$, then using $\lfloor \phi_{ijf}(t) \rfloor$ is feasible. Q.E.D.

The value of Proposition 1 is that we no longer have to be concerned about whether variables $\phi_{ijf}(t)$ are integer or not, i.e. branching on variables $h^r_{\pi f}$ is sufficient to find a feasible integer solution to problem (25).

After solving problem (26), we may obtain many $h^r_{\pi f}$ values in the interval $(0,1)$. These values are obviously not feasible for problem (25). A simple method to attempt to obtain a feasible solution is to try rounding $h^r_{\pi f} - \epsilon$ for some small number $\epsilon > 0$. The ϵ is added because it is possible to have $h^r_{\pi_1 f} = h^r_{\pi_2 f} = 0.5$, which would obviously be infeasible if both are rounded to 1. It is possible for this forced rounding to violate constraint (25c) in general. However, if all capacities are equal to 1, then we can prove that rounding $h^r_{\pi f}$ results in a feasible solution to problem (26). Although this assumption is limiting, it is not completely unreasonable. Because UAVs travel in the air, it is possible for the capacities of links of equal width to be equal. If so, since C_{if} is typically in units of drones per unit time, the assumption $C_{if} = 1$ could be made true by choosing an appropriate time step and by splitting links into parallel links of equal width. However, this would also increases the number of variables for problem (20), and the number of potential columns for BP. Regardless, Proposition 2 is still useful for building intuition about the problem structure.

PROPOSITION 2. *Let $\lfloor h \rceil$ indicate the nearest integer function, i.e. rounding h to the nearest integer. Let $\epsilon > 0$ be a very small positive number. Suppose that $C_{if} = 1$ for all $i \in \mathcal{A}$ and $f \in \mathcal{F}$. After solving problem (26) to obtain $h^r_{\pi f}$, the solution $\left\lfloor h^r_{\pi f} - \epsilon \right\rceil$ is a feasible solution to problem (25).*

Proof. We first note that for any set of numbers $h_\pi \geq 0$ satisfying $\sum_\pi h_\pi < 1$, $\sum_\pi \lfloor h_\pi \rceil \leq 1$. Given $h^r_{\pi f}$ satisfying constraint (25b), $\sum_{\pi \in \Pi_r} \sum_{f \in \mathcal{F}_r} \left(h^r_{\pi f} - \epsilon\right) < 1$, so $\sum_{\pi \in \Pi_r} \sum_{f \in \mathcal{F}_r} \left\lfloor h^r_{\pi f} - \epsilon \right\rceil \leq 1$ satisfies constraint (25b). Similarly for constraint (25c), $\sum_{r \in \mathcal{R}} \sum_{\pi \in \Pi_r} \delta^\pi_i(t) \left(h^r_{\pi f} - \epsilon\right) < C_{if} = 1$ by assumption, so $\sum_{r \in \mathcal{R}} \sum_{\pi \in \Pi_r} \delta^\pi_i(t) \left\lfloor h^r_{\pi f} - \epsilon \right\rceil \leq C_{if} = 1$. Constraint (25d) is replaced by $\phi_{ijf} \geq \sum_{\pi \in \Pi_r} \delta^\pi_{ij}(t) \left\lfloor h^r_{\pi f} - \epsilon \right\rceil$. Because of constraint (25e), for all sets of conflicting turning movements, at most one (i,j) will have $\phi_{ijf}(t) - \epsilon \geq 0.5$. Therefore, after replacing $h^r_{\pi f}$ with $\left\lfloor h^r_{\pi f} - \epsilon \right\rceil$, there still exists a vector of $\phi_{ijf}(t)$ such that constraint (25e) holds. Q.E.D.

Even if $C_{if} \neq 1$ for some links or flight levels, Proposition 2 still establishes some useful intuition about a rounding heuristic. Rounding yields a solution that satisfies constraints (25b), (25d), and (25e). Therefore, one way to construct a feasible solution after solving linear program (26) is to round $h^r_{\pi f} \in (0.5, 1)$ to 1 as long as constraint (25c) is satisfied, then round fractional h to 0 afterwards. This procedure is implemented as a heuristic within the proposed BP algorithm.

3.3. Column generation

After solving problem (26) with the restricted sets of columns $\bar{\Pi}_r$, the CG algorithm aims to find new columns that may improve the objective function. In our study, the CG pricing subproblem is that of finding time-dependent paths in a time-expanded network which is a directed acylic graph. Since the RMP is a maximization problem, the CG pricing subproblem is that of finding longest time-dependent paths. From an algorithmic standpoint, it is more convenient to work with path-cost minimization formulations. Since the path-finding problem is solved in a directed acyclic graph, it is equivalent to look for the shortest time-dependent path in a graph with negated link weights. Hence, in the CG pricing subproblem we negate all coefficients of variables' reduced costs expressions and seek the time-dependent shortest path. Let λ_r be the dual variable of constraint (25b), let $\mu_{if}(t)$ be the dual variable of constraint (25c) and let $\theta_{ijf}(t)$ be the dual variable of constraint (25d). The (negated) reduced cost of variable $h^r_{\pi f}$, denoted $c^r_{\pi f}$, is:

$$c^r_{\pi f} = -\alpha p_r + (1-\alpha)\tau_{\pi f} + \lambda_r + \sum_{i \in \mathcal{A}} \sum_{t=0}^{T} \delta^\pi_i(t) \mu_{if}(t) + \sum_{(i,j) \in \mathcal{A}^2} \sum_{t=0}^{T} \delta^\pi_{ij}(t) \theta_{ijf}(t) \tag{27}$$

A path variable $h^r_{\pi f}$ is said to have a negative reduced-cost if $c^r_{\pi f} < 0$. The goal of CG pricing is to find negative reduced-cost variables. If upon solving the RMP there are no negative reduced-cost variables, then the LP solution is optimal. Hence, to prove optimality we need to solve:

$$\bar{c} = \min_{r \in \mathcal{R}, \pi \in \Pi_r, f \in \mathcal{F}_\pi} \left\{ c^r_{\pi f} \right\} \tag{28}$$

If $\bar{c} \leq 0$, then there are no negative reduced-cost variable left and we have solved the linear program (26) to optimality. Otherwise, if $\bar{c} > 0$, then there exists at least one column that might increase the objective value Z. To solve the pricing subproblem (28), it is natural to decompose it into request-based subproblems. Thus, for each request $r \in \mathcal{R}$, we aim to find the pair of path and flight level $(\pi_r^\star, f_r^\star)$ such that:

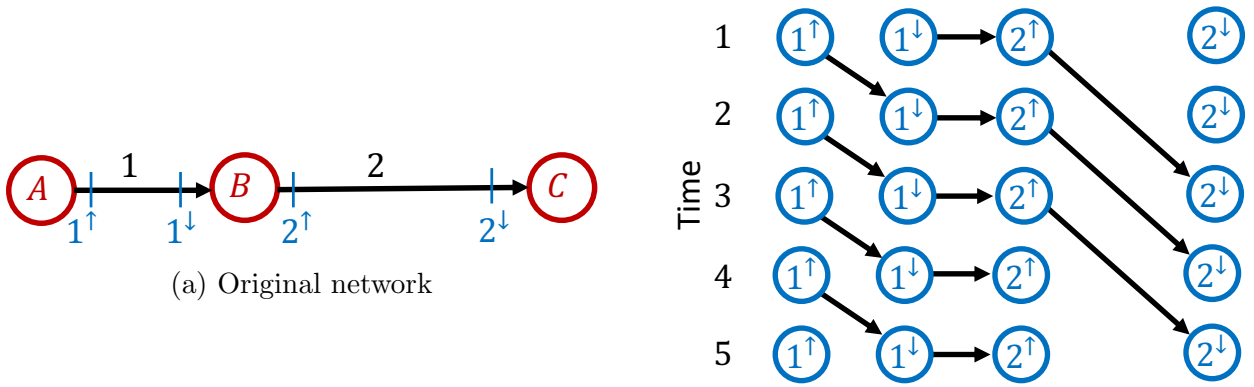
$$(\pi_r^\star, f_r^\star) \in \underset{\pi \in \Pi_r, f \in \mathcal{F}_\pi}{\arg\min} \left\{ c_{\pi f}^r \right\} \tag{29}$$

and we add variable $h^r_{\pi_r^\star, f_r^\star}$ to $\bar{\Pi}_r$ if $c^r_{\pi_r^\star f_r^\star} > 0$. In the next section, we present an efficient algorithm to solve the pricing subproblem (29). At each CG iteration, up to $|\mathcal{R}|$ variables may be added. Therefore, we also consider removing columns from the restricted path sets $\bar{\Pi}_r$ to keep the number of variables smaller. We implement this column removal by keeping the n columns with the highest $c^r_{\pi f}$ values, as well as keeping any columns used in the optimal solution.

3.4. Time-dependent shortest path

Both equations (28) and (29) require finding the shortest path, or the path with the minimum reduced cost. Fortunately, we can exploit the problem structure to find the shortest path for each request in linear time. Consider the time-expanded dual graph $\mathcal{G}^{\mathrm{T}}$ where nodes are tuples (i,t) where $i \in \mathcal{A}$ and $t \in \{0, \ldots, T\}$. $\mathcal{G}^{\mathrm{T}}$ is the dual graph of $\mathcal{G}$ in that the nodes of $\mathcal{G}^{\mathrm{T}}$ are based on the links of $\mathcal{G}$. Links in $\mathcal{G}^{\mathrm{T}}$ connect node (i,t) with node $\left(j, t + \frac{\ell_i}{v_f}\right)$, which represents entering link i at time t and traveling to link j. Link $\left[(i,t), \left(j, t + \frac{\ell_i}{v_f}\right)\right]$ is associated with constraint (25c) for link i at time t, and with constraint (25d) for $\phi_{ijf}\left(t + \frac{\ell_i}{v_f}\right)$. Therefore, the reduced cost of link $\left[(i,t), \left(j, t + \frac{\ell_i}{v_f}\right)\right]$ is $-(1-\alpha)\frac{\ell_i}{v_f} - \mu_{if}(t) - \theta_{ijf}\left(t + \frac{\ell_i}{v_f}\right)$. To facilitate finding optimal departure times, we also add nodes (z,t) associated with zone $z \in \mathcal{Z}$ at time $t \in \{0, \ldots, T\}$. We add links $[(z,t),(i,t)]$ for all $i \in \Gamma_z^+$ to indicate departure onto link i. We further add links $[(z,t),(z,t+1)]$ indicating that departure is delayed to $t+1$ or later. These added links have 0 reduced cost.

Figure 1 illustrates the conversion from the urban airspace network to the time-expanded dual graph. Figure 1a shows three nodes, A, B, and C, connected by two links. Link 1 has a travel time of one time step and link 2 has a travel time of two time steps. The dual graph includes nodes at both the upstream and downstream ends of each link, i.e. $1^\uparrow$, $1^\downarrow$, $2^\uparrow$, and $2^\downarrow$. These nodes are expanded in time as shown in Figure 1b. Links between $1^\uparrow$ to $1^\downarrow$ span one time step, which is the travel time of link 1. Links between $2^\uparrow$ to $2^\downarrow$ span two time steps, which is the travel time of link 2. There are also links from $1^\downarrow$ to $1^\uparrow$, which occur without a time delay since movement across an intersection is modeled as occurring within one time step. Travel from node A to node C, departing at $t=1$, requires three time steps for an arrival at node C at $t=4$; this is equivalent to the path consisting of nodes $(1^\uparrow,1)$, $(1^\downarrow,2)$, $(2^\uparrow,2)$, and $(2^\downarrow,4)$.



(a) Original network

(b) Time-expanded network

Figure 1 **Illustration of the conversion to a time-expanded graph**

Because links always connect forward in time due to travel times, network $\mathcal{G}^{\mathrm{T}}$ is a directed acyclic graph. Although the reduced costs of links will change at each CG iteration, the network structure itself will not change. Therefore, $\mathcal{G}^{\mathrm{T}}$ can be topologically sorted once upon network construction, and reduced costs can be updated later. Finding the shortest time-dependent path in this directed acyclic graph requires $\mathcal{O}(|\mathcal{A}| \times |\mathcal{F}| \times T)$ time. This procedure is used to solve the pricing subproblem (29) for all requests $r \in \mathcal{R}$ at each iteration of the CG algorithm.

3.5. Branching rules

As shown by Proposition 1, it is sufficient to branch on path variables $h^r_{\pi f}$ to find integer solutions and extensive numerical experiments have revealed that branching on the most fractional $h^r_{\pi f}$ values yields a balanced performance. Yet, it is well-known that in BP algorithms for set partitioning formulations branching on the variables of the compact formulation is more efficient than branching on path-based variables (Ropke and Cordeau 2009, Vanderbeck 2011). To this end, we also consider additional branching rules that are intended to provide an alternative to branching on fractional $h^r_{\pi f}$ variables. Observe that path-based variables $h^r_{\pi f}$ can be linked to variables z_r using

$$z_r = \sum_{\pi \in \Pi_r} \sum_{f \in \mathcal{F}_r} h^r_{\pi f} \tag{30}$$

Let $[h^{r\star}_{\pi f}]$ be the current solution of the RMP (26) and let $[z^\star_r]$ be the corresponding z-solution obtained using Eq. (30). Consider the following branching rules.

Rule 1: Branch on the number of accepted requests. If $\sum_{r \in \mathcal{R}} z^\star_r$ is fractional, then generate two children nodes with the cuts:

$$\sum_{r \in \mathcal{R}} \sum_{\pi \in \Pi_r} \sum_{f \in \mathcal{F}} h^r_{\pi f} \leq \left\lfloor \sum_{r \in \mathcal{R}} z^\star_r \right\rfloor \quad \text{and} \quad \sum_{r \in \mathcal{R}} \sum_{\pi \in \Pi_r} \sum_{f \in \mathcal{F}} h^r_{\pi f} \geq \left\lceil \sum_{r \in \mathcal{R}} z^\star_r \right\rceil .$$

Rule 2: Branch on a request acceptance variable. If there exists a request $r \in \mathcal{R}$ such that variable $z_r^\star$ is fractional, then generate two children nodes with the cuts:

$$\sum_{\pi \in \Pi_r} \sum_{f \in \mathcal{F}} h_{\pi f}^r \leq \lfloor z_r^\star \rfloor \text{ and } \sum_{\pi \in \Pi_r} \sum_{f \in \mathcal{F}} h_{\pi f}^r \geq \lceil z_r^\star \rceil.$$

Rule 3: Branch on the set of FLs. If there exists a request $r \in \mathcal{R}$ and a pair of FLs $f < f'$ such that: $\sum_{\pi \in \Pi_r} h_{\pi f}^{r\star} > 0$ and $\sum_{\pi \in \Pi_r} h_{\pi f'}^{r\star} > 0$, then we can partition the set of FLs $\mathcal{F}_r$ such that f and f' belong to different partitions. Let $\mathcal{F}_r^<$ and $\mathcal{F}_r^>$ be those partitions, i.e. $\mathcal{F}_r^< \cup \mathcal{F}_r^> = \mathcal{F}_r$, $\mathcal{F}_r^< \cap \mathcal{F}_r^> = \varnothing$, $f \in \mathcal{F}_r^<$ and $f' \in \mathcal{F}_r^>$. Then, generate two children nodes with the cuts:

$$\sum_{\pi \in \Pi_r} \sum_{f \in \mathcal{F}_r^<} h_{\pi f}^r = 0 \text{ and } \sum_{\pi \in \Pi_r} \sum_{f \in \mathcal{F}_r^>} h_{\pi f}^r = 0.$$

In our numerical experiments, we compare the performance of two configurations for the proposed BP algorithm: i) branching solely on the most fractional $h_{\pi f}^r$ values, and ii) applying the above three branching rules before resorting to branching on the most fractional $h_{\pi f}^r$ values. We next present two heuristics that are used to generate feasible integer solutions at the beginning and during the BP algorithm.

3.6. Reservation heuristic

Inspired by Waller and Ziliaskopoulos (2006) and Levin (2019), the structure of the time-expanded dual graph $\mathcal{G}^{\mathrm{T}}$ can be exploited to find and then "reserve" paths that are compatible with constraints (25c)–(25e). The approach works by removing connectivity in $\mathcal{G}^{\mathrm{T}}$ when constraints (25c) or (25e) would be violated. Specifically, when $\sum_{r \in \mathcal{R}} \sum_{\pi \in \Pi_r} \delta_i^\pi(t) h_{\pi f}^r > C_{if} - 1$, then link $\left[(i,t), \left(j, t + \frac{\ell_i}{v_f}\right)\right]$, for any $j \in \Gamma_i^+$, is marked as disconnected. Similarly, if $\phi_{ijf}(t) = 1$, then for any $(i',j') \in \mathcal{C}_{ij}$, link $\left[\left(i, t - \frac{\ell_i}{v_f}\right), (j,t)\right]$ is marked as disconnected. Let π be the shortest path in $\mathcal{G}^{\mathrm{T}}$ calculated while ignoring disconnected links for request r and flight level f. Then it is possible to set $h_{\pi f}^r = 1$ while keeping the values of $h_{\pi f}^{r'}$ for all requests $r' \neq r$, and the resulting solution will still satisfy constraints (25c)–(25e).

It is possible that after connectivity is removed, no path exists for request r, which prevents this approach from being optimal. However, this still yields an efficient reservation heuristic that can be used to generate an initial feasible solution and set of columns. Iterate through the set $\mathcal{R}$ in some order. For each r, find the longest available path-flight level tuple $(\pi_r^\star, f)$ in $\mathcal{G}^{\mathrm{T}}$. Then "reserve" that path by setting $h_{\pi_r^\star f}^r = 1$ and breaking connectivity accordingly. The resulting solution provides at most one path for each request, and is guaranteed to be feasible. The reservation heuristic is described in Algorithm 1. Notice that in line 3, the arg max is taken over all possible paths $\pi \in \Pi_r$, not over all paths in the restricted set $\bar{\Pi}_r$. In Section 4, we will show that for simple problems with few conflicts, this heuristic is often capable of finding an optimal solution very quickly.

Algorithm 1 Reservation heuristic

for $r \in \mathcal{R}$ **do**
  if $\nexists \pi \in \Pi_r, f \in \mathcal{F}_\pi$ s.t. $h^r_{\pi f} = 1$ **then**
    Find $(\pi^\star_r, f^\star_r) \in \underset{\pi \in \Pi_r, f \in \mathcal{F}_\pi}{\arg\max} \left\{c^r_{\pi f}\right\}$ satisfying (25c)–(25e)
    Break connectivity in constraints (25c)–(25e) for $\pi^\star_r$
  end if
end for

3.7. Primal heuristic

In practice, we found that the upper bound obtained from the CG algorithm was often not a strong upper bound until many cuts were applied. This can be observed in results in Section 4 showing the upper bound and lower bound with respect to the BP node. Frequently, the upper bound did not decrease much, and the lower bound was much lower than the upper bound. To more quickly find a competitive lower bound, we developed a primal heuristic which adds many cuts to search similar BP nodes with near-integer solutions.

Let $\mathcal{B}$ be the set of BP nodes and let $b \in \mathcal{B}$ represent a node in the BP tree. We denote $\mathcal{C}_b$ the set of cuts at b, where each cut is a constraint specifying that $h^r_{\pi f} = 1$ or $h^r_{\pi f} = 0$ for a specific request r, path π, and flight level f. Instead of branching on a single fractional $h^r_{\pi f}$ variable, we define $\mathcal{H}^{(0.5,1)}_b$ to be the set of $h^r_{\pi f}$ variables with values $h^r_{\pi f}$ in the interval $(0.5, 1)$. Further define $\mathcal{H}^{[1]}_b$ to be the set of $h^r_{\pi f}$ variables with values $h^r_{\pi f} = 1$. We define two sets of additional cuts $\mathcal{C}^{(0.5,1)}_b = \left\{h^r_{\pi f} = 1 : h^r_{\pi f} \in \mathcal{H}^{(0.5,1)}_b, \forall r \in \mathcal{R}, \pi \in \bar{\Pi}_r, f \in \mathcal{F}_\pi\right\}$ and $\mathcal{C}^{[1]}_b = \left\{h^r_{\pi f} = 1 : h^r_{\pi f} \in \mathcal{H}^{[1]}_b, \forall r \in \mathcal{R}, \pi \in \bar{\Pi}_r, f \in \mathcal{F}_\pi\right\}$. The proposed primal heuristic consists of creating additional branches from a given BP node with cuts from $\mathcal{C}^{(0.5,1)}_b$ and $\mathcal{C}^{[1]}_b$. Specifically, we create multiple extra branches b' with

$$\mathcal{C}_{b'} = \mathcal{C}_b \cup \mathcal{C}^{[1]}_b \cup \hat{\mathcal{C}}^{(0.5,1)}_b(b') \tag{31}$$

Each of these branches include the cuts $\mathcal{C}_b$, all of the cuts in $\mathcal{C}^{[1]}_b$, as well as *some* of the cuts in $\mathcal{C}^{(0.5,1)}_b$. The set $\hat{\mathcal{C}}^{(0.5,1)}_b(b') \subseteq \mathcal{C}^{(0.5,1)}_b$ denotes which cuts from $\mathcal{C}^{(0.5,1)}_b$ are included in branch b'. If we include more cuts, the solution is less optimal but closer to being integer. We therefore consider multiple branches with different $\hat{\mathcal{C}}^{(0.5,1)}_b(b')$ sets. We define subsets with $\left\lfloor \frac{|\mathcal{C}^{(0.5,1)}_b|}{1} \right\rceil$, $\left\lfloor \frac{|\mathcal{C}^{(0.5,1)}_b|}{2} \right\rceil$, $\left\lfloor \frac{|\mathcal{C}^{(0.5,1)}_b|}{4} \right\rceil$, etc. of the cuts in $\mathcal{C}^{(0.5,1)}_b$, and create a new branch b' for each of these subsets.

This procedure is specified in Algorithm 2, and is incorporated as an optional heuristic to the proposed BP algorithm. Overall, this primal heuristic generates many more BP nodes that will more quickly lead to integer solutions. In the worst case, where $h^r_{\pi f} \in (0.5, 1)$ do not exist, the

primal heuristic devolves to standard BP. Therefore, this heuristic changes the order in which some BP nodes are evaluated while still guaranteeing optimality.

Algorithm 2 Primal heuristic (supplements branch creation in Algorithm 3 just before line 17)

1: $\mathcal{H}_b^{(0.5,1)} \leftarrow \left\{h_{\pi f}^r : h_{\pi f}^r \in (0.5,1), \forall r \in \mathcal{R}, \pi \in \bar{\Pi}_r, f \in \mathcal{F}_\pi\right\}$
2: $\mathcal{H}_b^{[1]} \leftarrow \left\{h_{\pi f}^r : h_{\pi f}^r = 1, \forall r \in \mathcal{R}, \pi \in \bar{\Pi}_r, f \in \mathcal{F}_\pi\right\}$
3: $\xi \leftarrow 1$
4: **while** $\left\lfloor \frac{|\mathcal{C}_b^{(0.5,1)}|}{\xi} \right\rfloor > 0$ **do**
5: $\quad \hat{\mathcal{C}}_b^{(0.5,1)} \leftarrow$ first ξ elements of $\left\{h_{\pi f}^r = 1 : h_{\pi f}^r \in \mathcal{H}^{(0.5,1)}, \forall r \in \mathcal{R}, \pi \in \bar{\Pi}_r, f \in \mathcal{F}_\pi\right\}$ sorted by $h_{\pi f}^r$
6: $\quad \mathcal{C}_b^{[1]} \leftarrow \left\{h_{\pi f}^r = 1 : h_{\pi f}^r \in \mathcal{H}^{[1]}, \forall r \in \mathcal{R}, \pi \in \bar{\Pi}_r, f \in \mathcal{F}_\pi\right\}$
7: $\quad \mathcal{C}_{b'} = \mathcal{C}_b \cup \hat{\mathcal{C}}_b^{(0.5,1)} \cup \mathcal{C}_b^{[1]}$
8: $\quad \overline{Z}_{b'} \leftarrow \overline{Z}_b$
9: $\quad \underline{Z}_{b'} \leftarrow \underline{Z}_b$
10: $\quad \mathcal{B} \leftarrow \mathcal{B} \cup \{b'\}$
11: $\quad \xi \leftarrow 2\xi$
12: **end while**

3.8. Overview of the branch-and-price algorithm

At each BP node $b \in \mathcal{B}$, we denote the upper bound on the objective $\overline{Z}_b$ and the lower bound $\underline{Z}_b$. The upper bound is obtained by the CG procedure and the lower bound is obtained by one of the heuristics discussed above to convert the CG solution into a feasible integer solution. Among all solved nodes, we retain the best lower bound $\underline{Z}^\star$, which is the best feasible solution found so far. The optimality gap g is defined as the percent difference between the maximum upper bound of among BP nodes and the best (maximal) lower bound:

$$g = \frac{\max\limits_{b \in \mathcal{B}} \left\{\overline{Z}_b\right\} - \underline{Z}^\star}{\underline{Z}^\star} \tag{32}$$

The BP algorithm stops if the set of active nodes $\mathcal{B}$ is empty or if the optimality gap is smaller than some stopping criteria. The pseudocode of the proposed BP algorithm is presented in Algorithm 3.

As shown in line 9, the reservation heuristic is used at the root node of the BP tree to find an initial feasible solution and lower bound. The CG procedure is then used to solve the (26) at the current node. If an integer solution is not found, then in line 15 we use a rounding heuristic to attempt to construct an integer feasible solution. Some requests may not be assigned to a path. For those requests, we try to improve the lower bound through the reservation heuristic in line 16.

Algorithm 3 BP algorithm for drone delivery service planning

1: $\mathcal{B} \leftarrow \{0\}$
2: $\underline{Z}^\star \leftarrow -\infty$
3: **while** $\mathcal{B} \neq \varnothing$ and $g > \underline{g}$ **do**
4: Remove b from $\mathcal{B}$
5: **if** $\overline{Z}_b \leq \underline{Z}^\star$ **then**
6: **continue**
7: **end if**
8: **if** $b = 0$ **then**
9: $\underline{Z}_b \leftarrow$RESERVATIONS()
10: **end if**
11: $\overline{Z}_b \leftarrow$COLUMN-GENERATION($\mathcal{C}_b$)
12: **if** $h^r_{\pi f} \in \{0,1\}$ for all $r \in \mathcal{R}$, $\pi \in \Pi_r$, $f \in \mathcal{F}_\pi$ **then**
13: $\underline{Z}_b \leftarrow \overline{Z}_b$
14: **else**
15: $\underline{Z}_b \leftarrow$ROUNDING($\mathbf{h}$)
16: $\underline{Z}_b \leftarrow$RESERVATIONS($\mathbf{h}$)
17: Find $h^r_{\pi f} \in (0,1)$
18: $\mathcal{C}_{b'} \leftarrow \mathcal{C}_b \cup \left\{h^r_{\pi f} = 1\right\}$
19: $\mathcal{C}_{b''} \leftarrow \mathcal{C}_b \cup \left\{h^r_{\pi f} = 0\right\}$
20: $\overline{Z}_{b'} \leftarrow \overline{Z}_{b''} \leftarrow \overline{Z}_b$
21: $\underline{Z}_{b'} \leftarrow \underline{Z}_{b''} \leftarrow \underline{Z}_b$
22: $\mathcal{B} \leftarrow \mathcal{B} \cup \{b', b''\}$
23: **end if**
24: $\underline{Z}^\star \leftarrow \max\left\{\underline{Z}^\star, \underline{Z}_b\right\}$
25: $g \leftarrow \frac{\max_{b \in \mathcal{B}}\left\{\overline{Z}_b\right\} - \underline{Z}^\star}{\underline{Z}^\star}$
26: **end while**

Then, lines 17–22 construct the two new branches based on the fractional $h^r_{\pi f}$ that is chosen. The loop continues while the gap g is greater than the stopping criteria $\underline{g}$.

There are several options for the ordering of steps in the BP algorithm that affect the best feasible solution found after some given time. In the reservation heuristic (Algorithm 1), the order in which requests are processed determines which paths are available to requests that are processed later. We sorted requests by their earliest departure time, e_r. We explored randomizing the order of

requests, but few differences were observed. This may occur because sorting by earliest departure time already randomizes the selection of origin-destination pairs to some extent.

In addition, the order in which BP nodes are evaluated can have a significant impact on the computation time. Because BP nodes with $\overline{Z}_b < \underline{Z}^\star$ can be discarded immediately, increasing $\underline{Z}^\star$ as much as possible early on can eliminate many BP nodes. As demonstrated in Section 4, we observed that finding a good lower bound was often limited by having many fractional $h^r_{\pi f}$ values. For that reason, we opt for a depth first search approach, which explores BP nodes with the most cuts first. However, evaluating branch nodes with a small value of $\overline{Z}_b$ may find a solution that is guaranteed to not be optimal.

4. Numerical results

The purpose of these numerical results is to demonstrate the performance of the proposed BP algorithm on realistic networks. We have already alluded to some of the performance gains from the heuristics proposed in Sections 3.6 and 3.7, and the numerical performance of these heuristics is shown here. We used IBM CPLEX 12.10 to solve all linear and integer linear programs, i.e. within the CG procedure and for solving the ILP of problem (20). For clarity, we will hereafter refer to problem (20) as the ILP solution method in contrast with the BP algorithm proposed in Section 3. All results were obtained on a laptop computer with an Intel Core i7-1065G7 processor at 1.3GHz with 16GB of memory. All memory was made available to CPLEX. We also evaluate the BP algorithm against directly solving the ILP. As will be shown, the number of variables in the ILP quickly becomes limiting for realistic problems. Even where the data is small enough for CPLEX to directly solve the ILP, the performance of the BP algorithm is usually reasonably similar.

Obtaining real data for numerical experiments is not easy. Datasets exist for freight delivery are often focused on vehicle routing problems and only include travel times relevant to the specified requests. Since network capacity limitations and intersection conflicts are an important part of this problem and contribute to its complexity, such datasets provide insufficient information to construct the urban network. Since we assume that the UAV airspace exists above road networks, we can make use of extensive data on traffic networks, including road lengths and intersection locations. Unfortunately, the demand specified in these datasets are usually for person-trips or private vehicle trips. The trip tables are probably not representative of typical UAV trips in terms of the origin-destination demand. Nevertheless, traffic network datasets provide more information on the urban network, which extends to UAV airspace. We have therefore used these datasets to describe realistic networks, but adjusted link capacities and speeds to reflect UAV characteristics. The trip patterns are still based on private vehicle trips, but because these trip patterns are known to cause congestion in the road network, these patterns should also create an interesting

challenge for UAV trajectory optimization. Discrete requests were generated randomly proportional to exogenous trip patterns. Since arrival time windows are not specified, we randomly generated arrival time windows as follows. The earliest arrival time is the earliest departure time plus the free flow time. The latest arrival time is between 2–15 minutes after the earliest arrival time, with the specific value randomly chosen for each trip request.

These experiments are conducted on the Sioux Falls network, which is a standard dataset for traffic assignment problems. The network has 24 nodes (which are also zones) and 76 links. Unlike the traffic assignment version, capacities were set to 60 UAVs/hr. These capacities are likely stricter than necessary to avoid collisions between UAVs. However, the size of the ILP was restricted by the number of variables, and using small capacities admits solving the problem on a larger network with active capacity constraints. The earliest departure time of the demand was drawn from a uniform random distribution over 15min. The departure time window was set to 15 minutes for each request, and arrival time windows were drawn from a uniform random distribution over $[2, 13]$min. 1 hour was given for UAVs to exit. The time step was 1 minute.

Table 1 Comparison of BP methods with respect to increasing $|\mathcal{R}|$ and $|\mathcal{F}|$ for the profit balance scenario ($\alpha = 0.9$, $p_r = 10$)

(a) Performance metrics

		ILP			BP-primal heuristic				BP				BP-branching rules				Reservation		
$\|\mathcal{R}\|$	$\|\mathcal{F}\|$	obj.	gp.	rt.	LB	gp.	rt.	it.	LB	gp.	rt.	it.	LB	gp.	rt.	it.	LB	gp.*	rt.
200	1	1591.1	0.01	27.5	1590.8	0.03	17.9	5	1590.6	0.00	103.5	37	1591.2	0.00	18	5	1567.2	1.53	0.25
300	1	2392.3	0.01	112.7	2346	1.99	3602.0†	978	2370.1	0.95	3602.1†	896	2322.8	3.01	3602.1†	771	2247.2	6.46	0.26
400	1	3174.9	0.02	3616.6†	3096	2.32	3602.6†	594	2957.2	7.40	3600.7†	563	2956.8	7.42	3603.0†	368	2880.3	10.23	0.38
500	1	3905.3	1.49	3640.8†	3687.1	7.12	3949.7†	451	3409.1‡	16.13	3603.0†	189	3409.1‡	16.13	3604.2†	121	3409.1	14.56	0.44
500	2		–		3967.4	0.19	3603.4†	385	3974.7	0.00	2537.3	188	3927.9‡	1.19	3601.5†	237	3927.9	1.19	0.89
600	2		–		4734.5	0.73	3605.7†	318	4664.2‡	2.25	3607.7†	203	4664.2‡	2.25	3605.8†	185	4664.2	1.51	1.01
700	2		–		5476.7	1.65	3607.6†	234	5360.1‡	3.86	3602.4†	160	5360.1‡	3.86	3614.9†	132	5360.1	2.18	1.08
800	2		–		6139.2	3.55	3607.2†	171	6001.2‡	5.93	3617.6†	106	6001.2‡	5.93	3600.4†	96	6001.2	2.30	1.22
700	3		–		5567.0	0.00	330.3	18	5544.1‡	0.41	3620.2†	112	5544.1‡	0.77	3620.4†	94	5544.1	0.41	2.11
800	3		–		6347.8	0.15	3611.0†	169	6308.3‡	0.77	3626.0†	83	6308.3‡	0.77	3628.1†	72	6308.3	0.70	2.22
900	3		–		7097.2	0.78	3617.3†	142	7050.8‡	1.45	3631.6†	60	7050.8‡	1.45	3622.4†	55	7050.8	0.66	2.26
1000	3		–		7838.3	1.37	3609.7†	125	7714.8‡	2.99	3643.6†	46	7714.8‡	2.99	3632.4†	44	7714.8	1.60	2.52

obj. is the objective value returned by CPLEX. LB is the best integer solution found using BP.
gp. is the optimality gap in %. *reported gap for the reservation heuristic is the difference relative to the best solution found.
rt. is the runtime in seconds. it. is the number of branches explored by BP.
†indicates termination due to the time limit of 3600s
‡indicates that BP was not able to find a better integer solution than the initial LB from reservations
"–" indicates insufficient computer memory to solve problem (ILP only)

(b) Computation times and column generation solution to the relaxed RMP

		Root node	CPU times (s) — BP-primal heuristic					CPU times (s) — BP					CPU times (s) — BP-branching rules				
$\|\mathcal{R}\|$	$\|\mathcal{F}\|$	obj. UB	reserve	price	TDSP	CPLEX	BP	reserve	price	TDSP	CPLEX	BP	reserve	pric	TDSP	CPLEX	BP
200	1	1591.2	4.5	0.4	1.5	10.7	0.6	31.3	2.1	8.5	57.4	2.6	3.8	0.5	1.7	11.4	0.4
300	1	2392.7	1177.1	3.6	10.4	2249.4	88.3	1015.8	53.0	188.5	2150.0	123.5	964.5	52.6	178.7	2237.6	108.2
400	1	3174.9	889.8	10.5	32.3	2526.7	82.5	768.9	54.1	195.7	2407.3	110.7	610.8	71.0	229.9	2577.4	72.4
500	1	3952.0	734.0	10.3	35.8	2671.1	86.1	297.2	71.2	271.8	2886.4	45.8	227.8	68.6	225.8	3029.1	31.8
500	2	3974.8	1553.8	1.5	5.5	1969.3	36.2	904.4	33.4	116.0	1415.1	42.8	1162.7	50.5	178.4	2117.0	58.4
600	2	4769.2	1509.2	2.7	11.1	2004.6	42.4	1101.9	48.7	177.7	2182.0	61.6	1005.7	52.1	176.2	2282.7	55.9
700	2	5566.9	1267.1	4.0	15.2	2243.8	43.3	915.0	44.1	148.1	2408.9	55.1	783.5	46.2	153.4	2559.0	46.0
800	2	6357.0	1094.6	6.3	25.0	2398.6	47.6	646.6	37.6	126.7	2742.5	40.9	592.0	39.5	138.8	2771.2	37.4
700	3	5567	160.3	0.9	4.2	159.2	3.4	1304.6	54.9	232.2	1962.9	41.4	1172.2	50.5	198.9	2144.4	34.5
800	3	6357.1	1617.1	5.0	22.3	1916.6	27.8	1032.7	48.4	191.6	2299.1	34.6	966.1	45.2	185.4	2385.2	29.4
900	3	7152.7	1586.7	6.0	28.1	1942.7	31.4	780.6	38.7	161.3	2609.7	26.5	718.5	36.5	134.6	2695.1	24.0
1000	3	7945.3	1471.5	3.8	17.5	2067.3	27.6	623.7	34.0	145.2	2807.5	21.2	595.2	30.3	116.3	2858.7	20.5

Table 2 Comparison of BP methods with respect to increasing $|\mathcal{R}|$ and $|\mathcal{F}|$ for the maximum served scenario ($\alpha = 1$, $p_r = 1$)

(a) Performance metrics

		ILP			BP-primal heuristic				BP				BP-branching rules				Reservation		
$\|\mathcal{R}\|$	$\|\mathcal{F}\|$	obj.	gp.	rt.	LB	gp.	rt.	it.	LB	gp.	rt.	it.	LB	gp.	rt.	it.	LB	gp.*	rt.
200	1	200	0.00	29.7	199	0.50	13.1	3	200	0.00	13.6	5	200	0.00	15.7	2	197	1.52	0.2
300	1	300	0.00	76.7	300	0.00	858.3	232	300	0.00	110.5	36	300	0.0	280.8	60	284	4.63	0.28
400	1	396	1.01	3610.7†	390	2.56	3604.4†	544	373	7.24	3604.3†	381	367	8.99	3603.7†	394	365	8.49	0.36
500	1	485	3.09	3602.5†	468	6.84	3601.5†	439	433‡	15.47	3623.8†	60	433‡	15.47	3606.6†	60	433	12.01	0.41
500	2		–		500	0.00	24.7	2	499‡	0.2	3625.3†	76	500	0.00	119.7	12	499	0.20	0.84
600	2		–		600	0.00	264.3	20	595‡	0.84	3706.2†	38	595‡	0.84	3962.2†	33	595	0.84	0.98
700	2		–		700	0.00	1022.9	66	690‡	1.45	3687.2†	14	690‡	1.45	4042.1†	15	690	1.45	1.06
800	2		–		788	1.52	3609.5†	191	767‡	4.30	4035.1†	10	767‡	4.30	3634.5†	9	767	2.74	1.2
700	3		–		700	0.00	21.3	1	700‡	0.00	3942.1†	16	700‡	0.00	4139.6†	13	700	0.00	1.98
800	3		–		800	0.00	161.8	8	800‡	0.00	3787.9†	11	800‡	0.00	3967.4†	10	800	0.00	2.1
900	3		–		900	0.00	145.3	5	896‡	0.45	3627.7†	8	896‡	0.45	4232.6†	8	896	0.45	2.36
1000	3		–		1000	0.00	464.1	16	992‡	0.81	4376.1†	7	992‡	0.81	5103.3†	6	992	0.81	2.54

obj. is the objective value returned by CPLEX. LB is the best integer solution found using BP.
gp. is the optimality gap in %. *reported gap for the reservation heuristic is the difference relative to the best solution found.
rt. is the runtime in seconds. it. is the number of branches explored by BP.
†indicates termination due to the time limit of 3600s
‡indicates that BP was not able to find a better integer solution than the initial LB from reservations
"–" indicates insufficient computer memory to solve problem (ILP only)

(b) Computation times and column generation solution to the relaxed RMP

		Root node	CPU times (s) — BP-primal heuristic					CPU times (s) — BP					CPU times (s) — BP-branching rules				
$\|\mathcal{R}\|$	$\|\mathcal{F}\|$	obj. UB	reserve	price	TDSP	CPLEX	BP	reserve	price	TDSP	CPLEX	BP	reserve	price	TDSP	CPLEX	BP
200	1	200.0	3.0	0.3	1.4	7.8	0.3	3.2	0.3	1.3	8.3	0.2	1.7	0.2	0.9	4.7	0.1
300	1	300.0	264.3	1.2	3.4	551.7	21.3	30.7	1.7	5.7	67.6	3.0	76.4	3.8	10.6	176.1	9.0
400	1	400.0	884.8	10.0	23.4	2535.3	84.0	669.8	42.0	118.7	2632.4	91.8	647.8	43.7	127.8	2653.8	85.1
500	1	499.0	864.2	8.0	18.5	2545.8	97.7	127.2	9.2	22.4	3435.6	18.9	111.5	8.2	21.7	3441.9	14.9
500	2	500.0	9.0	0.4	1.9	12.7	0.4	687.6	26.6	76.0	2786.3	32.1	42.1	2.3	10.6	62.5	1.3
600	2	600.0	92.5	0.7	3.0	163.0	3.1	335.5	15.6	48.1	3285.6	13.9	210.7	9.0	29.5	3698.6	9.4
700	2	700.0	360.0	2.4	9.5	630.2	12.8	115.3	6.8	22.3	3535.3	4.9	111.6	5.2	17.5	3899.6	5.2
800	2	800.0	1289.6	3.8	14.2	2236.6	38.7	75.0	3.9	14.4	3936.2	3.6	80.2	5.4	18.2	3525.5	3.4
700	3	700.0	9.8	0.4	1.9	9.1	0.0	287.8	10.6	40.3	3595.3	5.1	158.8	6.5	28.5	3939.8	3.8
800	3	800.0	70.3	0.6	2.8	85.2	2.1	145.1	6.2	25.8	3605.9	3.1	128.6	5.4	25.2	3802.8	3.4
900	3	900.0	48.2	1.8	9.1	84.3	1.4	101.4	4.1	19.2	3499.5	2.2	142.9	5.9	24.0	4054.5	3.2
1000	3	1000.0	162.1	3.4	16.5	275.2	4.8	91.1	3.9	18.4	4259.4	2.0	128.4	7.6	36.6	4926.8	2.3

4.1. Comparing BP with ILP

Our first set of experiments are focused on evaluating the performance of the proposed BP algorithm on the path-based formulation (25) compared with a direct ILP solve of the link-based formulation (20). The initial results assumed a single flight level, again due to memory limitations in the ILP. On the computer used, the ILP was observed to run out of memory at around 650 requests for a single flight level. With multiple flight levels, the number of variables and network capacity both increase linearly. Using multiple flight levels would result in an uncongested network.

Tables 1 and 2 give the objective values reached for selected instances, and Figures 2 and 5 plot more objective values. Both CPLEX and BP were given a time limit of 3600s. In places where the computation time for BP exceeds 3600s, this is due to CPLEX requiring a long time to solve the restricted master problem (26) as part of CG. In those cases, CG and BP were terminated after CPLEX returned, and the solution was not updated. Such instances are marked with a low number of explored branches, indicating their high computation time per branch. The algorithms were terminated after reaching a gap of less than 1E-2. As $|\mathcal{F}|$ increases, the network capacity also increases necessitating an increase in the number of requests to create an interesting scenario. Most of the instances reported in Tables 1 and 2 are difficult to solve to the point that BP terminated after the time limit. This is because, as illustrated later in Figures 2 and 5, easier problems tended to have a feasible solution with the maximum possible objective value.

Table 1 corresponds to the profit balance scenario where $\alpha = 0.9$ and $p_r = 10$ for all requests $r \in \mathcal{R}$. Table 2 corresponds to the case $\alpha = 1$ and $p_r = 1$ for all requests $r \in \mathcal{R}$ which simply tries to maximize the number of requests served. We set the convergence criterion $\underline{g} = 1e^{-2}$ and use a time limit of 1 hour. The label "ILP" denotes solving problem (20). "BP-primal heuristic" refers to BP with the primal heuristic with BP nodes evaluated in depth-first search order. "BP" refers to BP without the primal heuristic and prioritizing BP nodes by their upper bound, i.e. best-first search. We use "branching rules" to refer to BP with the branching rules defined in Section 3.5, and BP nodes are also solved in order of highest upper bound. Although BP-primal heuristic with best-first search is not shown, it was observed to perform slightly worse than BP-primal heuristic with depth-first search. The optimality gaps shown in Tables 1a and 2a are based on the best upper bounds found by that algorithm (which varied for each algorithm) and were calculated using equation (32). The reservation heuristic does not attempt to find an upper bound, so the optimality gap listed for it is the difference between the best found solution and the solution obtained from the reservation heuristic. If the problem is sufficiently easy, such a feasible solution could be found by the reservation heuristic alone.

We observe several patterns of note in Tables 1a and 2a. First, the ILP solved by CPLEX consistently found the best solution when the time limit was reached. However, it did not have

sufficient memory to solve any nontrivial instances with $|\mathcal{F}| > 1$. Second, for easy instances, i.e. $|\mathcal{R}| = 300$ with $|\mathcal{F}| = 1$, BP and BP-branching rules found the optimal solution much faster than BP-primal heuristic. This is likely attributed to the order in which branch nodes are visited.

For the more difficult instances reported in Tables 1 and 2, i.e. with two or more flight levels, BP and BP-branching rules performed much worse than BP-primal heuristic in finding a good integer feasible solution. This may be partly due to the number of BP nodes evaluated. BP-primal heuristic also sought an integer feasible solution more aggressively than BP or BP-branching rules. Although that solution was often not optimal, it was still often better than the best integer solution found after rounding the non-integer solution from BP or BP-branching rules. These limitations became more acute with increasing $|\mathcal{F}|$.

The reservation heuristic is included as a separate column in Tables 1a and 2a as a comparison. It is much faster to execute than any of the BP methods. For most of the scenarios, BP and BP-branching rules never found a better integer solution than the reservation heuristic used in line 9. This occurred because in trying to improve the objective value, CG resulted in many non-integer values for $h^r_{\pi f}$. In contrast, BP-primal heuristic was consistently able to improve on the reservation heuristic. These results suggest that the reservation heuristic is preferable to BP and BP-branching rules unless the time limit is very long. For instances with a relatively large number of requests and with a correspondingly large number of flight levels, the CG procedure took a significant amount of time. This is shown by the small number of BP nodes evaluated in Tables 1a and 2a for instances with three flight levels for example. Although one might consider these large scenarios to be unrealistic, the performance of the reservation heuristic suggests that smaller numbers of requests could be solved in seconds without using BP at all. This is true even with capacities of 1 drone per minute per link, which are likely lower than necessary. Overall, the general conclusion is that a good solution to most easy problems can be quickly obtained by the reservation heuristic, and the primal heuristic but not BP or BP-branching rules consistently creates an improved integer solution within a reasonable time limit.

CPLEX was unable to solve a problem with 350 UAVs and 2 flight levels on a network with 24 nodes and 76 links. We expect most realistic cities to be much larger, and 350 UAVs on 2 flight levels is insufficient for congestion. Therefore, even if a more powerful computer were to be used, it would be easy to find realistic problems in major cities for which the memory is insufficient. The number of variables and constraints in the link-based formulation are both $O(|\mathcal{A}| \times |\mathcal{F}| \times |\mathcal{R}| \times T)$. In contrast, the structure of the branch-and-price algorithm means that the number of variables is $O(|\mathcal{R}| \times |\mathcal{F}|)$, assuming a bound on the number of paths stored per UAV flight request. The number of constraints is $O(|\mathcal{A}| \times T)$. The above analysis suggests that increasing computer memory would

not be sufficient for solving the link-based formulation for a problem with 350 UAVs, 4 flight levels, and 304 links, which is still a far smaller network than expected for major cities.

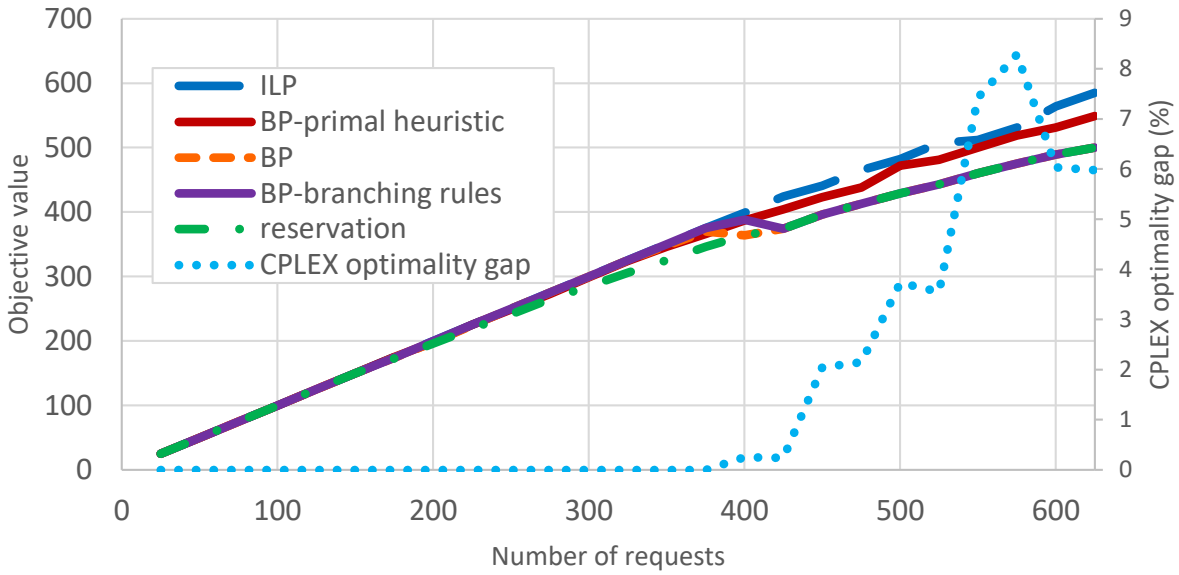
Tables 1b and 2b also show the objective value achieved by the root node of CG (without any cuts). For most instances, CG was able to achieve an upper bound on the objective value computed by calculating the minimum travel time for each request, disregarding all congestion constraints. Often, the solution used by CG to obtain this upper bound was non-integer.

The following sections discuss the results in more detail, using plots for a more visual comparison. They are roughly organized around different scenarios, e.g. $|\mathcal{F}| = 1$ with $\alpha = 1$ and $p_r = 1$, or $|\mathcal{F}| = 1$ with $\alpha = 0.9$ and $p_r = 10$.

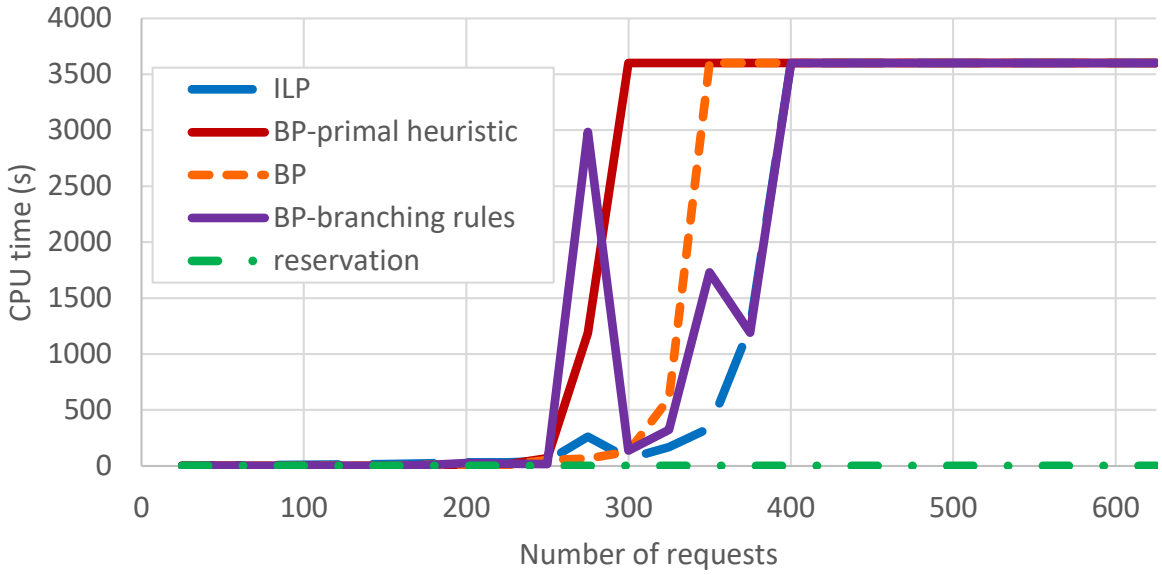
4.1.1. Maximize number of served requests We first explore the performance for the objective of maximizing the number of served requests ($\alpha = 1, p_r = 1$). In Figure 2, the objective values are plotted against the number of requests at increments of 25 to show how increasing numbers of requests affects the difficulty in solving the problem. ILP consistently found the best solution, but the solution found by BP-primal heuristic was very close, always within 6.2% difference. In contrast, branching on $h^r_{\pi f}$ variables without the primal heuristic often did not improve over the reservation heuristic. The branching rules even performed worse on large problems. These results suggest that the solution algorithm used by CPLEX is quite well optimized for this problem. Figure 2 also reports the optimality gap reported by CPLEX after 1 hr of computation time. The benefits of BP are its ability to solve much larger problems than can be solved with the ILP. BP with or without the branching rules often did not improve on the reservation heuristic. We also studied the primal heuristic in combination with the branching rules; the branching rules became active when all remaining $h^r_{\pi f}$ variables had values in $(0, 0.5)$. This was not observed to perform much different from the primal heuristic without the branching rules.

Computation times were quite similar except for the region between 275–375 requests, in which CPLEX terminates far faster than the BP methods. This is partly due to the difficulty in proving optimality from BP. BP usually had difficulty reducing the upper bound; when only a few cuts are present, the upper bound was observed to not decrease much through using fractional values of $h^r_{\pi f}$. However, if a faster solution is desired, the reservation heuristic could find a good solution in only a few seconds. This solution is plotted in Figure 2a, and is not much worse than BP with the primal heuristic. In fact, the reservation heuristic frequently performed as well as BP without the primal heuristic. (BP used the reservation heuristic to construct the initial feasible solution.)

Figure 3 and Tables 1b and 2b compare the computation times used for different components of the BP algorithm. Most of the computation time goes towards solving linear programs as part of column generation (labeled as "CPLEX" time). The second significant use of time is for the reservation heuristic, which is used both to find an initial feasible solution and after column generation



(a) Objective value



(b) Computation time

Figure 2 **Comparison of BP and ILP methods on Sioux Falls,** $\alpha = 1$ **and** $p_r = 1$

completes to improve the integer solution. Note that the results shown in Figure 2 from the reservation heuristic only come from the first iteration, whereas the computation times attributed to "reservation" in Figure 3 include running the reservation heuristic each iteration of BP to improve the lower bound in line 16 of Algorithm 3. The computation time for the time-dependent shortest path (TDSP) is quite short, as it is easy to execute on a directed acyclic graph. The reservation heuristic is more complex than TDLP due to the need to update connectivity and find paths

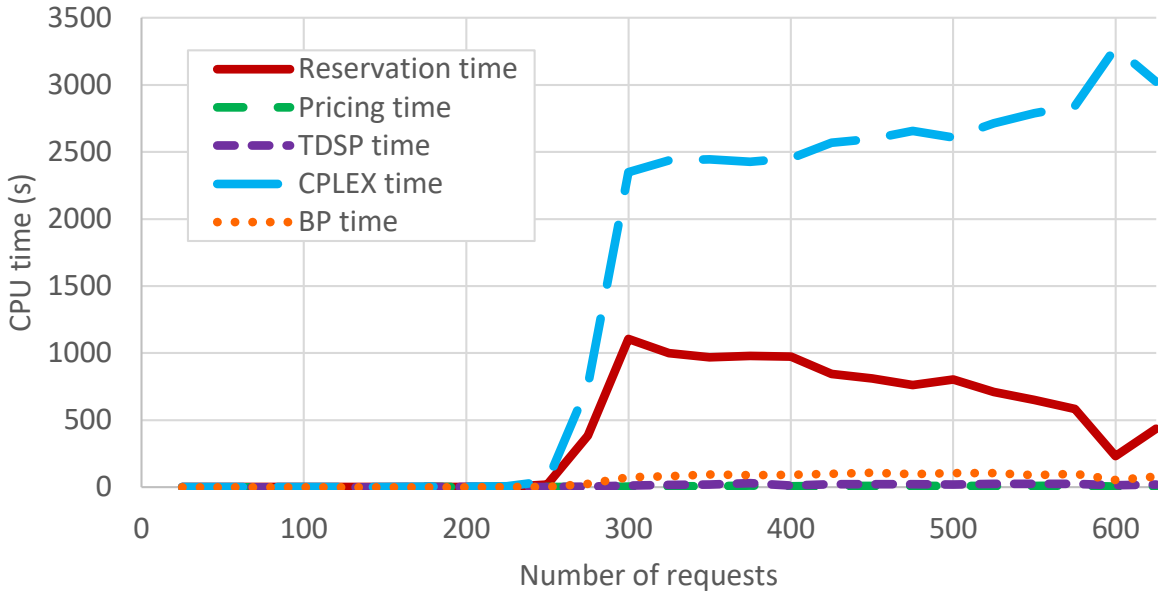


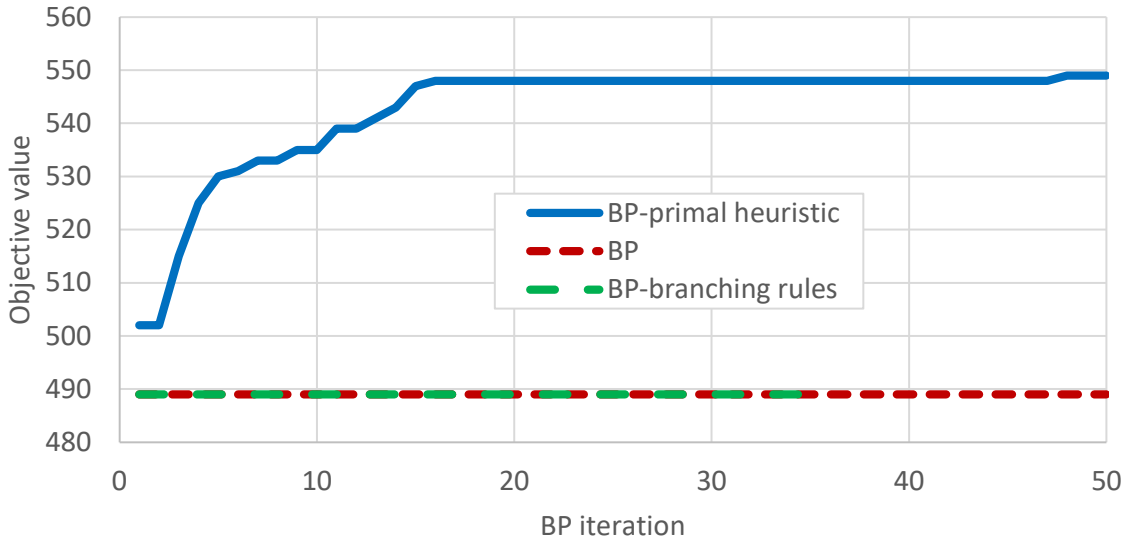
Figure 3 **Computation time for different components of BP-primal heuristic on Sioux Falls, $\alpha = 1$ and $p_r = 1$. "CPLEX time" refers to time spent solving the restricted master problem for column generation using CPLEX.**

that are connected in terms of not violating constraints. The time spent on pricing paths and on constructing branches is minimal.

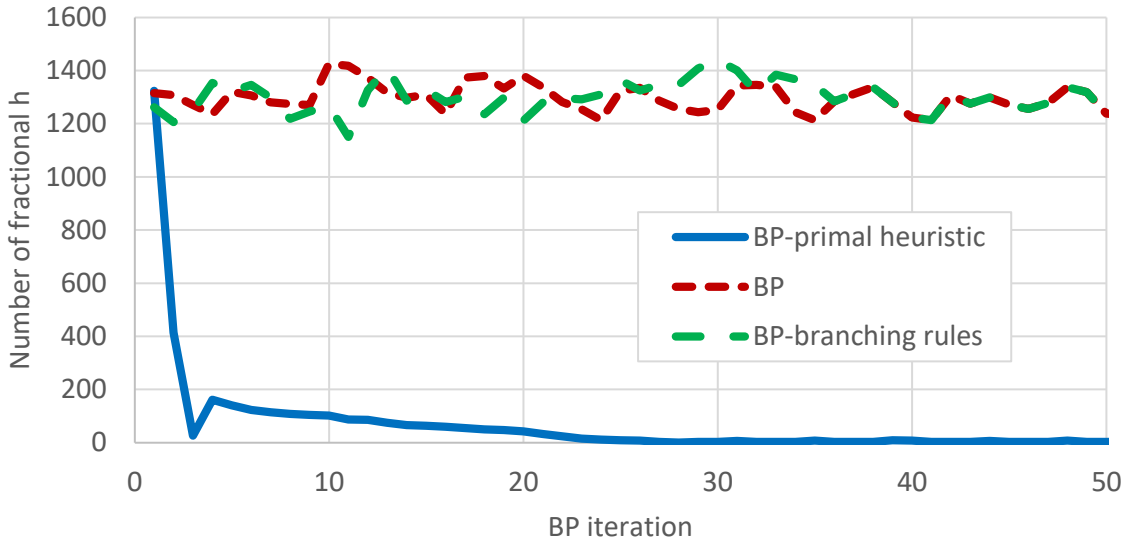
To illustrate the performance of BP, Figure 4 characterizes the solution output of BP per iteration with 500 requests. Each iteration corresponds to one branch node being evaluated. Figure 4a shows the best feasible solution found for the first 50 iterations of BP. Most of the improvement occurs early in BP, with only small improvements occurring later. This appears to be a tradeoff with the primal heuristic. By prioritizing the local search, significant improvement is achieved early around that initial feasible solution, but BP stays near that initial solution for the entire 1 hour of runtime. BP with the branching rules appeared to improve more steadily than the other versions of BP, but far fewer iterations were executed due to higher computation time per iteration.

Figure 4b shows the number of $h^r_{\pi f} \in (0,1)$ values for the first 50 BP iterations, which must be rounded to achieve a feasible solution. The number of fractional $h^r_{\pi f}$ values is initially quite high, but decreases rapidly for the primal heuristic. However, it remains much higher for BP without the primal heuristic. The number of column generation iterations was fairly small after the first iteration of BP. However, the large number of cuts made by the primal heuristic resulted in much lower computation times per column generation and much fewer fractional $h^r_{\pi f}$ values. The output of primal heuristic was therefore easier to round to a good integer feasible solution.

4.1.2. Profit balance objective The profit balance objective balances the number of served requests with the total travel time. The objective function parameters were chosen as $\alpha = 0.9$ and



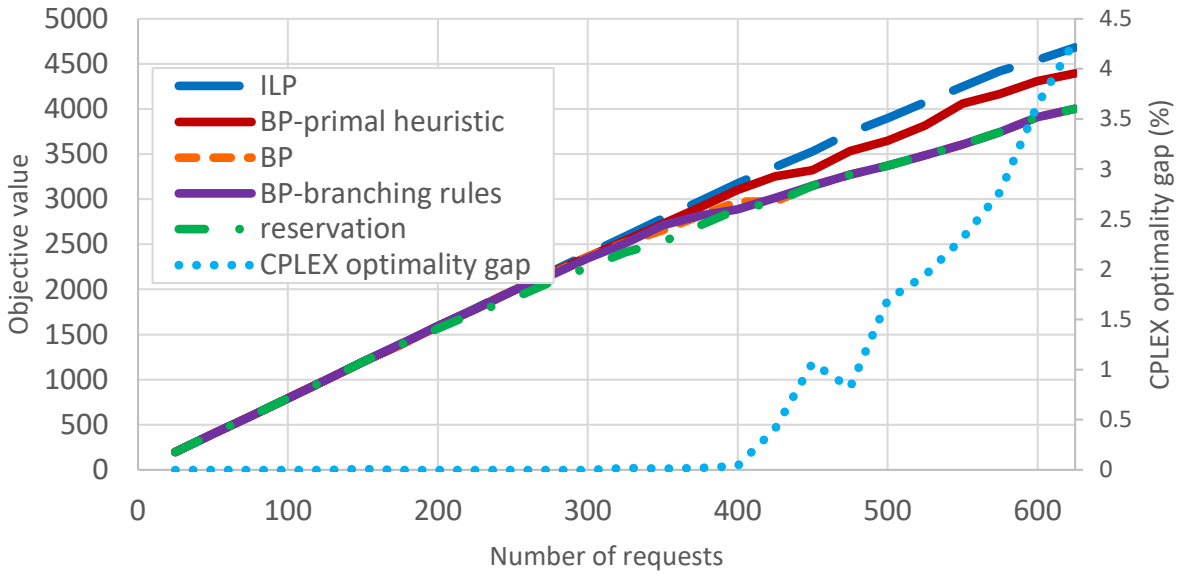
(a) Objective value



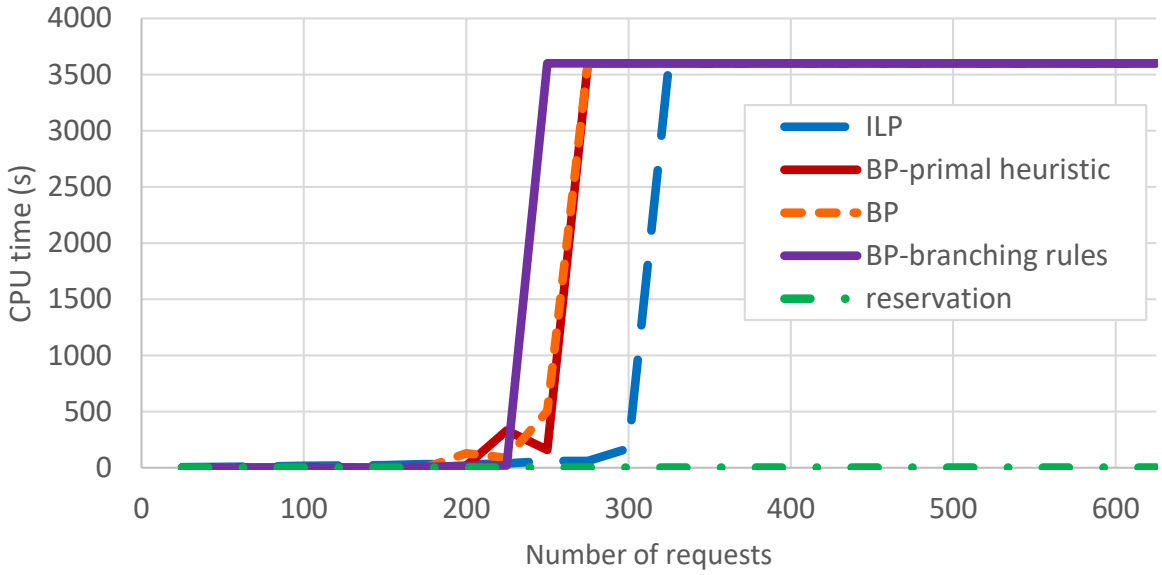
(b) Fractional $h^r_{\pi f}$ values

Figure 4 **Performance of BP with primal heuristic on Sioux Falls with 600 requests, $\alpha = 1$ and $p_r = 1$. The best**

$p_r = 10$ for all requests. These parameter values were used because they provide positive value for serving a request, but also provide an incentive to minimize travel time. The problem appears more challenging to solve with the profit balance objective. When comparing the results of Tables 1a and 2a, the profit balance objective tended to result in higher runtimes or larger gaps at termination. Similar patterns to Section 4.1.1 are observed in Figure 5. BP with the primal heuristic performs similarly to the ILP, and in several cases actually found a better solution within the time limit of 1 hr of computation. The reservation heuristic by itself was able to quickly solve simple problems.



(a) Objective value



(b) Computation time

Figure 5 **Comparison of BP and ILP methods on Sioux Falls,** $\alpha = 0.9$ **and** $p_r = 10$

When the number of requests was 200 or less, the reservation heuristic resulted in a significant time reduction over the ILP. Figure 5 also compares the reservation heuristic with BP. With a large number of requests, BP without the primal heuristic was unable to find a better integer solution than the reservation heuristic. BP-branching rules again performed better than branching on $h^r_{\pi f}$ variables, but worse than BP-primal heuristic.

4.2. Multiple flight levels

Next, we compare the different BP methods on the Sioux Falls network with more flight levels and more trip requests. Results from the ILP are not presented here because they exceeded the available memory on the computer used for testing. The scalability of BP itself is a valuable contribution as the straightforward ILP formulation is limited to small problems. These further comparisons of the different BP methods suggest that the primal heuristic is the best among them.

We first consider two flight levels, with the demand ranging from 600 to 1200 trips over 15 minutes. Tables 1 and 2 show selected instances for both objective functions. Like with one flight level, the primal heuristic with depth first search performed best. Surprisingly, branching on $h^r_{\pi f}$ or using the configuration BP-branching rules was unable to improve over the reservation heuristic. The reservation heuristic also performed fairly close to BP-primal heuristic which prioritizes finding an integer solution by adding multiple cuts. As discussed earlier, the primal heuristic may not be producing a tight upper bound quickly. A better upper bound would decrease the optimality gap. Only the primal heuristic appeared able to improve on the reservation heuristic.

Tables 1 and 2 also show instances with three flight levels and correspondingly larger demand. The same patterns appeared throughout. The primal heuristic was often able to achieve a much smaller gap than BP or BP with branching rules. For the scenarios with $|\mathcal{F}| = 2$ or $|\mathcal{F}| = 3$, the primal heuristic was even able to solve some instances before terminating due to the time limit, unlike BP or BP with branching rules.

4.3. Effects of increasing the time horizon

The number of variables and problem difficulty scale with the size of the time horizon, T. For our scenarios, we also define a subset of the time horizon, $T_{\mathcal{R}} < T$, which is the time horizon for the minimum departure time of requests. For the results in Tables 1 and 2, $T_{\mathcal{R}} = 15$min and $T = 60$min. We now explore the impacts of increasing $T_{\mathcal{R}}$ and T. As a baseline, we consider 600 requests over 15 minutes with $|\mathcal{F}| = 1$, which was chosen because it is fairly congested. As we increase $T_{\mathcal{R}}$, we correspondingly increased T and the number of requests. The scenarios and performance of BP are reported in Tables 3 and 4. We were only able to solve the link-based formulation for 600 requests over 15 minutes; we had insufficient computer memory to solve larger time horizons with more requests.

The time horizon results are consistent with the patterns observed in Tables 1 and 2, but also reveals new findings. First, extending the time horizon and requests quickly expands the memory requirements for the link-based formulation, and the 16GB of memory available to our computer was unusable for more than 600 requests over 15min. Both the BP and BP-branching rules methods performed poorly as the time horizon increased. This appears to be due to the high

computation times of CG resulting in only a few branch nodes being explored. Consequently, the reservation heuristic performed better, and its computation time was much lower. In contrast, BP-primal heuristic performed significantly better than the reservation heuristic itself, achieving 10–17% improvements. Overall, these results show that increasing the time horizon correspondingly increases the size of the problem and the difficulty in finding a good solution. Furthermore, they demonstrate that in congested airspace with a relatively high volume of requests, the BP-primal heuristic yields significant improvements over the reservation heuristic.

We also conducted time horizon experiments in which the departure time window is restricted to 5 minutes per request, instead of 15 minutes in the previous results. These results are reported in Tables 3b and 4b. Naturally, this made finding feasible path assignments for requests more difficult than in previous scenarios. Consequently, the reservation heuristic performed poorly compared to the BP-primal heuristic algorithm for these scenarios. Specifically, the solutions found by the reservation heuristic for the largest instance tested ($|\mathcal{R}| = 1200$) were reduced by 29.8% (Table 3b) and 24.9% (Table 4b) compared to BP-primal heuristic for the profit balance and maximum served scenarios, respectively. This behavior is driven by the fact that the reservation heuristic reserve paths for UAVs in a greedy manner which then prevented other UAVs from receiving path assignments at all.

Table 3 Effects of increasing time horizon on the profit balance scenario ($\alpha = 0.9$, $p_r = 10$) with $|\mathcal{F}|$=1

(a) 15min departure time windows

			ILP			BP-primal heuristic				BP				BP-branching rules				Reservation		
$\lvert\mathcal{R}\rvert$	T	$T_\mathcal{R}$	obj.	gp.	rt.	LB	gp.	rt.	it.	LB	gp.	rt.	it.	LB	gp.	rt.	it.	LB	gp.*	rt.
600	60	15	4542.8	3.64	3603.4†	4236.8	11.39	3606.0†	244	3512.8‡	34.41	3603.6†	70	3558.1‡	32.51	3609.2†	58	3916.3	16.00	0.43
800	80	20		–		5220.9	12.40	3612.0†	148	4231.3‡	38.47	3613.0†	21	4057.8‡	44.47	3608.7†	25	4631.1	12.74	0.83
1000	100	25		–		5956.2	12.60	3611.4†	95	4909.2‡	36.75	3606.9†	18	4624.9‡	45.35	3620.8†	16	5151.1	15.63	1.45
1200	120	30		–		6543.3	12.55	3614.5†	67	5105.2‡	44.67	3618.7†	18	4896.3‡	50.85	3611.9†	18	5555.5	17.82	2.4

(b) 5min departure time windows

			ILP			BP-primal heuristic				BP				BP-branching rules				Reservation		
$\lvert\mathcal{R}\rvert$	T	$T_\mathcal{R}$	obj.	gp.	rt.	LB	gp.	rt.	it.	LB	gp.	rt.	it.	LB	gp.	rt.	it.	LB	gp.*	rt.
600	60	15	4242.1	5.77	3608.5†	3995.5	13.99	3600.3†	283	3238.7‡	40.57	3602.5†	47	3085.7‡	47.73	3606.2†	47	3581.9	18.43	0.46
800	80	20		–		4818.5	13.39	3603.7†	128	3771.5‡	45.23	3607.4†	16	3651.1‡	49.77	3609.5†	19	4005.2	20.31	0.93
1000	100	25		–		5322.4	14.68	3609.4†	136	4358.3‡	40.38	3625.5†	16	4415.4	38.42	3612.0†	18	4312.1	23.43	1.58
1200	120	30		–		5853.7	13.34	3600.3†	74	5152.2	28.98	3623.9†	14	4851.1	36.74	3614.3†	17	4508.4	29.83	2.13

obj. is the objective value returned by CPLEX. LB is the best integer solution found using BP.
gp. is the optimality gap in %. *reported gap for the reservation heuristic is the difference relative to the best solution found.
rt. is the runtime in seconds. it. is the number of branches explored by BP.
T is the total time horizon available, but all requests have an initial departure time within $T_\mathcal{R}$. T and $T_\mathcal{R}$ are reported in minutes.
†indicates termination due to the time limit of 3600s
‡indicates that BP was not able to find a better integer solution than the initial LB from reservations
"–" indicates insufficient computer memory to solve problem (ILP only)

Table 4 Effects of increasing time horizon on the maximum served scenario ($\alpha = 1$, $p_r = 1$) with $|\mathcal{F}|$=1

(a) 15min departure time windows

			ILP			BP-primal heuristic				BP				BP-branching rules				Reservation		
$\|\mathcal{R}\|$	T	$T_{\mathcal{R}}$	obj.	gp.	rt.	LB	gp.	rt.	it.	LB	gp.	rt.	it.	LB	gp.	rt.	it.	LB	gp.*	rt.
600	60	15	564	6.98	3604.4†	530	13.02	3600.6†	235	411‡	45.89	3626.0†	10	310‡	92.90	3751.8†	6	487	15.81	0.41
800	80	20		–		647	15.20	3612.9†	85	469‡	58.94	3615.5†	19	411‡	80.78	3660.1†	8	578	11.94	0.83
1000	100	25		–		728	6.28	3602.4†	65	553‡	53.31	3624.6†	11	471‡	79.83	3731.2†	6	646	12.69	1.43
1200	120	30		–		814	13.6	3623.5†	35	592‡	56.26	3634.2†	11	500‡	84.80	3642.3†	3	698	16.62	2.62

(b) 5min departure time windows

			ILP			BP-primal heuristic				BP				BP-branching rules				Reservation		
$\|\mathcal{R}\|$	T	$T_{\mathcal{R}}$	obj.	gp.	rt.	LB	gp.	rt.	it.	LB	gp.	rt.	it.	LB	gp.	rt.	it.	LB	gp.*	rt.
600	60	15	534	8.15	3606.9†	510	14.40	3603.6†	190	379‡	54.22	3610.9†	39	339‡	72.22	3769.4†	5	459	16.34	0.42
800	80	20		–		602	16.61	3605.7†	146	450‡	56.15	3626.9†	17	431‡	62.88	3810.8†	10	508	18.50	0.83
1000	100	25		–		669	16.19	3613.8†	108	532‡	46.22	3608.7†	12	481‡	61.33	3611.8†	7	561	19.25	1.33
1200	120	30		–		727	15.43	3622.0†	70	596	41.03	3606.3†	12	460‡	49.64	3836.9†	6	582	24.91	2.10

obj. is the objective value returned by CPLEX. LB is the best integer solution found using BP.
gp. is the optimality gap in %. *reported gap for the reservation heuristic is the difference relative to the best solution found.
rt. is the runtime in seconds. it. is the number of branches explored by BP.
T is the total time horizon available, but all requests have an initial departure time within $T_{\mathcal{R}}$. T and $T_{\mathcal{R}}$ are reported in minutes.
†indicates termination due to the time limit of 3600s
‡indicates that BP was not able to find a better integer solution than the initial LB from reservations
"–" indicates insufficient computer memory to solve problem (ILP only)

4.4. Effects of constrained range on performance

UAVs often have limited range. For the link-based formulation, we can add constraints to enforce maximum travel distances or times. Let L_r be the maximum distance range for UAV request r. Then we can add the following constraint to formulation (20):

$$\sum_{i\in\mathcal{A}}\sum_{f\in\mathcal{F}_i}\sum_{t=0}^{T}\chi_{if}^{r\uparrow}(t)\ell_i \le L_r, \qquad \forall r\in\mathcal{R}. \tag{33}$$

For the path-based formulation, we can restrict Π_r to only include paths with a distance satisfying the L_r range constraint. Because we assume constant travel speeds, a constraint on maximum time aloft can be addressed in the same manner. However, BP relies on quickly finding the time-dependent shortest path in a directed acyclic graph. The cost of links includes both $\mu_{if}(t)$ and $\theta_{ijf}(t)$ as described in Equation (27), which is different from the range. Adding the range constraint therefore requires solving a constrained shortest path problem, which is NP-hard (Wang and Crowcroft 1996). Several constrained shortest path algorithms are available from the network optimization and vehicle routing problem literature (Beasley and Christofides 1989, Feillet et al. 2004, Ropke and Cordeau 2009). Adapting these algorithms within the proposed BP algorithms is non-trivial and would significantly impact computational performance since we repeatedly find the shortest path during each iteration of column generation. This problem could also be tackled by adapting a bi-objective shortest path algorithm (Duque, Lozano, and Medaglia 2015), but the output is not guaranteed to satisfy our range constraint unless we prioritize distance over minimizing reduced costs.

Another alternative is to use a k-shortest path algorithm when looking for minimal reduced-cost columns. If we iterate through those k returned paths in order of cost, the first of those paths that satisfies the range constraint will be the optimal constrained shortest path. If k is sufficiently high, we are guaranteed to find the optimal path. If not, then this is still a heuristic for the NP-hard constrained shortest path problem with efficient computation time. Nevertheless, this approach entails implementing a k-shortest path procedure and calibrating k which may be difficult to do in practice. Therefore, instead of finding exact solutions to the constrained shortest path problem, we propose a simple but fast heuristic.

Consider a single request r with its destination d_r and arrival time window $[l_r, u_r]$. At the termination of the time-dependent shortest path algorithm (Section 3.4), we will have found a path for r to arrive at d_r at every time $t\in[l_r, u_r]$. Because travel times within the network are constant, these paths must arrive exactly at $t\in[l_r, u_r]$ and not earlier. Therefore, these paths will likely be different in response to time-dependent congestion. We propose to scan this set of paths and check

if at least one meets the range constraint. Hence, for each request r, after finding these paths, we iterate through them and find the minimum reduced cost path that satisfies the range constraint. If no path satisfy the range constraint, then we terminate the pricing procedure without finding a path. This algorithm is obviously suboptimal because if the minimum reduced cost path for a request r arriving at a certain time t violates the range constraint, then we will no longer consider paths arriving at t for request r until the next iteration of column generation. However, we choose to use this heuristic due to the NP-hardness of the constrained shortest path problem and the computation times required for BP without the range constraint.

Tables 5 and 6 present the performance of the ILP, reservation heuristic, and BP methods when the range constraint is present, and can directly be compared with Tables 1a and 2a. Note that the gap of BP methods reported in Tables 5 and 6 may not be accurate because the upper bound is from column generation, which is solved using a heuristic for constrained shortest path. For instance, in Table 6 for the $|\mathcal{R}| = 200$ case, BP and BP-branching rules terminates with 195 or 196 requests served and a gap of 0.00 despite the ILP finding a solution with 197 requests served. However, overall the pattern of the results is quite similar to what was previously observed. The reservation heuristic usually performs better than BP and BP-branching rules, especially when the number of requests is larger and the computation time per iteration of column generation is high. The BP-primal heuristic performed best among the BP methods and reservation heuristic, but not as well as the ILP. However, the BP methods are usable for larger scenarios whereas the ILP would not have sufficient memory.

We observe that the range constraint reduces the objective value, and furthermore, the decrease in objective increases with respect to the number of requests. In other words, as the network congestion increases, fewer requests are able to be served. This is most obvious for the maximum served scenario. In Table 2a, as network congestion increases, UAVs are given longer routes to avoid congestion. However, the range constraint prevents some of those longer routes, resulting in fewer requests served in Table 6. The same behavior reduces the objective values for the profit balance scenario. We also note that the reservation heuristic has a higher gap with the range constraint. This could be caused by a combination of the heuristic used to solve the constrained shortest path and the increased difficulty in finding an optimal solution when the range constraint is added.

Table 5 **Comparison of BP methods with constrained range for the profit balance scenario ($\alpha = 0.9$, $p_r = 10$) with $\mathcal{F} = 1$**

	ILP			BP-primal heuristic				BP				BP-branching rules				Reservation		
$\lvert\mathcal{R}\rvert$	obj.	gp.	rt.	LB	gp.[§]	rt.	it.	LB	gp.[§]	rt.	it.	LB	gp.[§]	rt.	it.	LB	gp.[*]	rt.
200	1570.8	0.00	60.0	1569.2	0.01	424.5	155	1556.6	0.92	3601.3[†]	1071	1563.2	0.05	3362.3[†]	1189	1546.7	1.56	0.2
300	2361.6	0.00	137.3	2287.8	3.31	3600.8[†]	908	2202.5	7.29	3602.1[†]	568	2215.4	6.65	3600.1[†]	728	2188.1	7.93	0.22
400	3046.6	1.68	3608.7[†]	2863.8	8.31	3603.3[†]	598	2610.1	19.31	3602.4[†]	197	2627.4	18.39	3602.9[†]	202	2688.5	13.32	0.28
500	3632.0	3.47	3602.6[†]	3437.3	10.40	3602.4[†]	568	2922.6	30.47	3600.6[†]	93	2918.8	29.89	3604.7[†]	107	3150.3	15.29	0.37

obj. is the objective value returned by CPLEX. LB is the best integer solution found using BP.
gp. is the optimality gap in %. [*]reported gap for the reservation heuristic is the difference relative to the best solution found.
rt. is the runtime in seconds. it. is the number of branches explored by BP.
[†]indicates termination due to the time limit of 3600s
[‡]indicates that BP was not able to find a better integer solution than the initial LB from reservations
[§] indicates that the gap is derived from column generation using the heuristic for constrained shortest path
"–" indicates insufficient computer memory to solve problem (ILP only)

Table 6 **Comparison of BP methods with constrained range for the maximum served scenario ($\alpha = 1$, $p_r = 1$) with $\mathcal{F} = 1$**

	ILP			BP-primal heuristic				BP				BP-branching rules				Reservation		
$\lvert\mathcal{R}\rvert$	obj.	gp.	rt.	LB	gp.[§]	rt.	it.	LB	gp.[§]	rt.	it.	LB	gp.[§]	rt.	it.	LB	gp.[*]	rt.
200	197	0.00	34.2	195	0.26	26.1	7	195	0.00	4.6	1	196	0.00	2862.0	269	193	2.07	0.17
300	296	0.65	3601.7[†]	292	1.37	3601.4[†]	703	295	0.34	3604.0[†]	776	292	1.37	3601.8[†]	707	277	6.86	0.20
400	380	3.94	3603.0[†]	358	9.78	3603.0[†]	472	312	26.28	3600.8[†]	122	287	36.59	3625.0[†]	35	346	9.83	0.29
500	447	8.07	3601.5[†]	430	12.21	3606.3[†]	332	304	59.12	3613.4[†]	81	293	63.82	3632.3[†]	22	399	12.03	0.34

obj. is the objective value returned by CPLEX. LB is the best integer solution found using BP.
gp. is the optimality gap in %. [*]reported gap for the reservation heuristic is the difference relative to the best solution found.
rt. is the runtime in seconds. it. is the number of branches explored by BP.
[†]indicates termination due to the time limit of 3600s
[‡]indicates that BP was not able to find a better integer solution than the initial LB from reservations
[§]indicates that the gap is derived from column generation using the heuristic for constrained shortest path
"–" indicates insufficient computer memory to solve problem (ILP only)

4.5. Summary of numerical results

Overall, the main challenge in the BP algorithm appears to be finding an integer solution. Column generation seems to prefer non-integer solutions for increasing the objective value. Although each cut made progresses towards an integer solution, with hundreds or thousands of requests, there are many degrees of freedom for column generation to choose fractional values. By adding multiple cuts after each node, the primal heuristic creates integer solutions faster, although they are more likely to be suboptimal. However, given the computational time limit, searching for integer solutions faster appears to be more effective at improving the lower bound than rounding the fractional solutions otherwise generated. The primal heuristic is also able to search more branch nodes than the other BP methods because many of the branch nodes searched have a large number of cuts and are therefore much faster to evaluate using column generation.

The results in Tables 3 and 4 show that, compared to the reservation heuristic, the BP-primal heuristic performs up to 17.8% better in the profit balance scenario and up to 16.6% better in the maximum served scenario. If departure time windows are reduced to 5 minutes, the improvements are up to 29.8% in the profit balance scenario and 24.9% in the maximum served scenario. The reservation heuristic is a greedy assignment which iterates through the set of requests, ordered by their minimum departure time, and reserves the best remaining available path. If the network is congested, this greedy assignment is obviously suboptimal; it reserves airspace for individual requests that can more optimally be used in other ways. Therefore, we conclude that the reservation heuristic is effective at finding good solutions when the number of requests is small relative to the available airspace, but its optimality decreases as network congestion increases.

5. Conclusions

This study addresses a drone delivery service planning problem in urban airspace. We take the perspective of a UTM network manager which, given a set of drone delivery trip requests, aims to find the optimal assignment of drones to space-time trajectories in an urban air traffic network. The urban air traffic network is assumed to represent the airspace above an urban transportation network with multiple flight levels. We assume that drone deliveries must respect delivery time windows and that drones must fly at constant speed throughout the network to avoid airborne delays and congestion. We cast this drone delivery service problem as a dynamic network flow problem and we present link- and path-based mathematical optimization formulations. Our main contribution is to develop branch-and-price (BP) algorithms and customized heuristics to find (near-) optimal in competitive time. We further investigate three variations of BP: branching directly on $h^r_{\pi f}$ variables, branching on related variables, and a primal heuristic that adds extra cuts to find integer feasible solutions more quickly. We also propose a reservation heuristic that

quickly finds integer feasible solutions but rarely achieves optimality. Numerical results suggest that the primal heuristic is the most effective at problems with a time limit, and the reservation heuristic will often find a better integer solution than BP or BP with the branching rules.

The performance of a commercial solver (CPLEX) for solving the ILP (when sufficient computer resources are available) suggests that there are further opportunities to more efficiently solve this UTM approach. The performance of BP itself could be further improved, and other approaches for solving ILPs are also potentially relevant. The design of UTM airspace and its impact on drone delivery service planning is also worth exploring. This study focused on urban airspace with the assumption that it mostly exists above roads and can be described as a transportation network. Open air flights might benefit from different routing approaches. Finally, processing dynamic or stochastic demand, which is not fully known in advance, could be relevant for practical UAV trips.

Acknowledgments

References

Agatz N, Bouman P, Schmidt M, 2018 *Optimization approaches for the traveling salesman problem with drone. Transportation Science* 52(4):965–981.

Alejo D, Cobano J, Heredia G, Ollero A, 2013 *Particle swarm optimization for collision-free 4d trajectory planning in unmanned aerial vehicles. 2013 international conference on unmanned aircraft systems (ICUAS)*, 298–307 (IEEE).

Alharbi A, Poujade A, Malandrakis K, Petrunin I, Panagiotakopoulos D, Tsourdos A, 2020 *Rule-based conflict management for unmanned traffic management scenarios. 2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, 1–10 (IEEE).

Aurambout JP, Gkoumas K, Ciuffo B, 2019 *Last mile delivery by drones: an estimation of viable market potential and access to citizens across european cities. European Transport Research Review* 11(1):1–21.

Beasley JE, Christofides N, 1989 *An algorithm for the resource constrained shortest path problem. Networks* 19(4):379–394.

Bouman P, Agatz N, Schmidt M, 2018 *Dynamic programming approaches for the traveling salesman problem with drone. Networks* 72(4):528–542.

Boysen N, Schwerdfeger S, Weidinger F, 2018 *Scheduling last-mile deliveries with truck-based autonomous robots. European Journal of Operational Research* 271(3):1085–1099.

Carlsson JG, Song S, 2018 *Coordinated logistics with a truck and a drone. Management Science* 64(9):4052–4069.

Chamseddine A, Zhang Y, Rabbath CA, Theilliol D, 2012 *Trajectory planning and replanning strategies applied to a quadrotor unmanned aerial vehicle. Journal of Guidance, Control, and Dynamics* 35(5):1667–1671.

Chin C, Gopalakrishnan K, Egorov M, Evans A, Balakrishnan H, 2021 *Efficiency and fairness in unmanned air traffic flow management. IEEE Transactions on Intelligent Transportation Systems* .

Chin C, Gopalakrishnan K, Evans A, Egorov M, Balakrishnan H, 2020 *Tradeoffs between efficiency and fairness in unmanned aircraft systems traffic management. 9th International Conference on Research in Air Transportation.*

Cho J, Yoon Y, 2018 *How to assess the capacity of urban airspace: A topological approach using keep-in and keep-out geofence. Transportation Research Part C: Emerging Technologies* 92:137–149.

Coifman B, McCord M, Mishalani RG, Iswalt M, Ji Y, 2006 *Roadway traffic monitoring from an unmanned aerial vehicle. IEE Proceedings-Intelligent Transport Systems*, volume 153, 11–20 (IET).

Dias FH, Hijazi H, Rey D, 2022 *Disjunctive linear separation conditions and mixed-integer formulations for aircraft conflict resolution. European Journal of Operational Research* 296(2):520–538.

Dorling K, Heinrichs J, Messier GG, Magierowski S, 2016 *Vehicle routing problems for drone delivery. IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47(1):70–85.

D'Souza S, Ishihara A, Nikaido B, Hasseeb H, 2016 *Feasibility of varying geo-fence around an unmanned aircraft operation based on vehicle performance and wind. 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 1–10 (IEEE).

Duque D, Lozano L, Medaglia AL, 2015 *An exact method for the biobjective shortest path problem for large-scale road networks. European Journal of Operational Research* 242(3):788–797.

Feillet D, Dejax P, Gendreau M, Gueguen C, 2004 *An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks: An International Journal* 44(3):216–229.

Figliozzi MA, 2017 *Lifecycle modeling and assessment of unmanned aerial vehicles (drones) co2e emissions. Transportation Research Part D: Transport and Environment* 57:251–261.

Goodchild A, Toy J, 2018 *Delivery by drone: An evaluation of unmanned aerial vehicle technology in reducing co2 emissions in the delivery service industry. Transportation Research Part D: Transport and Environment* 61:58–67.

Ha QM, Deville Y, Pham QD, Hà MH, 2018 *On the min-cost traveling salesman problem with drone. Transportation Research Part C: Emerging Technologies* 86:597–621.

Ho F, Geraldes R, Goncalves A, Cavazza M, Prendinger H, 2018 *Improved conflict detection and resolution for service uavs in shared airspace. IEEE Transactions on Vehicular Technology* 68(2):1231–1242.

Ho F, Geraldes R, Goncalves A, Rigault B, Oosedo A, Cavazza M, Prendinger H, 2019 *Pre-flight conflict detection and resolution for uav integration in shared airspace: Sendai 2030 model case. IEEE Access* 7:170226–170237.

Huang S, Teo RSH, Tan KK, 2019 *Collision avoidance of multi unmanned aerial vehicles: A review. Annual Reviews in Control* 48:147–164.

Jenie YI, Van Kampen EJ, Ellerbroek J, Hoekstra JM, 2016 *Taxonomy of conflict detection and resolution approaches for unmanned aerial vehicle in an integrated airspace. IEEE Transactions on Intelligent Transportation Systems* 18(3):558–567.

Kirschstein T, 2020 *Comparison of energy demands of drone-based and ground-based parcel delivery services. Transportation Research Part D: Transport and Environment* 78:102209.

Klochkov VV, Karpov AE, 2018 *The prediction of transport-logistics systems based on unmanned aerial vehicles creation efficiency. 2018 Eleventh International Conference" Management of large-scale system development"(MLSD*, 1–5 (IEEE).

Kopardekar P, Rios J, Prevot T, Johnson M, Jung J, Robinson JE, 2016 *Unmanned aircraft system traffic management (utm) concept of operations. AIAA aviation forum.*

Kuru K, Ansell D, Khan W, Yetgin H, 2019 *Analysis and optimization of unmanned aerial vehicle swarms in logistics: An intelligent delivery platform. Ieee Access* 7:15804–15831.

Levin MW, 2019 *A combinatorial dynamic network trajectory reservation algorithm for connected autonomous vehicles. Networks and Spatial Economics* 19(1):27–55.

Mohamed Salleh MFB, Low KH, 2017 *Concept of operations (conops) for traffic management of unmanned aircraft systems (tm-uas) in urban environment. AIAA Information Systems-AIAA Infotech@ Aerospace*, 0223.

Mohamed Salleh MFB, Wanchao C, Wang Z, Huang S, Tan DY, Huang T, Low KH, 2018 *Preliminary concept of adaptive urban airspace management for unmanned aircraft operations. 2018 AIAA Information Systems-AIAA Infotech@ Aerospace*, 2260.

Murray CC, Chu AG, 2015 *The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. Transportation Research Part C: Emerging Technologies* 54:86–109.

NASA, 2021 *What is unmanned aircraft systems traffic management?* URL `https://www.nasa.gov/ames/utm`.

Ong HY, Kochenderfer MJ, 2017 *Markov decision process-based distributed conflict resolution for drone air traffic management. Journal of Guidance, Control, and Dynamics* 40(1):69–80.

Otto A, Agatz N, Campbell J, Golden B, Pesch E, 2018 *Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. Networks* 72(4):411–458.

Peinecke N, Kuenz A, 2017 *Deconflicting the urban drone airspace. 2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, 1–6 (IEEE).

Radzki G, Thibbotuwawa A, Bocewicz G, 2019 *Uavs flight routes optimization in changing weather conditions–constraint programming approach. Applied Computer Science* 15.

Ren L, Castillo-Effen M, Yu H, Yoon Y, Nakamura T, Johnson EN, Ippolito CA, 2017 *Small unmanned aircraft system (suas) trajectory modeling in support of uas traffic management (utm). 17th AIAA Aviation Technology, Integration, and Operations Conference*, 4268.

Rey D, Hijazi H, 2017 *Complex number formulation and convex relaxations for aircraft conflict resolution. 2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 88–93 (IEEE).

Rios J, Mulfinger D, Homola J, Venkatesan P, 2016 *Nasa uas traffic management national campaign: Operations across six uas test sites. 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, 1–6 (IEEE).

Ropke S, Cordeau JF, 2009 *Branch and cut and price for the pickup and delivery problem with time windows. Transportation Science* 43(3):267–286.

Sándor Z, 2019 *Challenges caused by the unmanned aerial vehicle in the air traffic management. Periodica Polytechnica Transportation Engineering* 47(2):96–105.

Toth P, Vigo D, 2001 *The vehicle routing problem* (Society for Industrial and Applied Mathematics).

Vanderbeck F, 2011 *Branching in branch-and-price: a generic scheme. Mathematical Programming* 130(2):249–294.

Vera S, Cobano JA, Heredia G, Ollero A, 2016 *Collision avoidance for multiple uavs using rolling-horizon policy. Journal of Intelligent & Robotic Systems* 84(1):387–396.

Waller ST, Ziliaskopoulos AK, 2006 *A combinatorial user optimal dynamic traffic assignment algorithm. Annals of Operations Research* 144(1):249–261.

Wang X, Poikonen S, Golden B, 2017 *The vehicle routing problem with drones: several worst-case results. Optimization Letters* 11(4):679–697.

Wang Z, Crowcroft J, 1996 *Quality-of-service routing for supporting multimedia applications. IEEE Journal on selected areas in communications* 14(7):1228–1234.

Wang Z, Sheu JB, 2019 *Vehicle routing problem with drones. Transportation research part B: methodological* 122:350–364.

Weibel R, Edwards M, Fernandes C, 2011 *Establishing a risk-based separation standard for unmanned aircraft self separation. 11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference, including the AIAA Balloon Systems Conference and 19th AIAA Lighter-Than*, 6921.

Yperman I, Logghe S, Immers B, 2005 *The link transmission model: An efficient implementation of the kinematic wave theory in traffic networks. Proceedings of the 10th EWGT Meeting, Poznan, Poland.*

Zhang X, Liu Y, Zhang Y, Guan X, Delahaye D, Tang L, 2018a *Safety assessment and risk estimation for unmanned aerial vehicles operating in national airspace system. Journal of Advanced Transportation* 2018.

Zhang Y, Zhai S, Wang D, Chen L, 2018b *Path planning-aiding system of unmanned aerial vehicle in freight transportation. 2018 International Conference on Network, Communication, Computer Engineering (NCCE 2018)*, 966–972 (Atlantis Press).