Optimal Location of Safety Landing Sites

Liding Xu

Advisor: Claudia D'Ambrosio, Leo Liberti, Sonia Vanier

July 15, 2021

Abstract

Vertical Take Off and Landing (VTOLs) vehichules are used to move passengers between skyports in urban air mobility. Safety landing sites (SLSs) cover the trajectory of VTOLs for emergency landings. We study the optimal placement of SLSs in the air transportation network under budget on SLS installation. We propose two models based on the k-splittable and the un-splittable multi-commodity flow problems. We develop edge and path formulations for each model of the problem. Edge formulations for two models are solved by a branch-and-bound algorithm. We propose a branch-and-price approach to solve path formulations. We perform numerical experiments on a set of automatically generated instances.

Keywords— Networks Optimization, Networks Simulation, Branch and Price, Multi-Commodity Flows, Column Generation

1 Introduction

The transportation system is getting more and more diverse and multimodality is a key factor. The current transportation network that helps us get from point A to point B by car, bike, scooter, and public transit—and, in the not-so-distant future, flights. Flying taxi (Vertical Take Off and Landing Vehicules: VTOLs) would exploit the vertical space i.e., to alleviate congestion on the ground and enable riders to click a button for a shared flight. Aerial ridesharing will enable rapid, reliable transportation through a network of small electric VTOLs (eVTOLs) that can take off and land vertically. We work in collaboration with Uber mobility team on energy efficient network design problem [HG16].

VTOL aircraft need to be safer than driving a car on a fatalities-per-passenger-mile basis. In emergence, VTOLs need a large number of spare landing sites i.e.

safety landing sites (SLS) to land. If the trajectory of a VTOL is not covered by ranges of installed SLSs, it would be dangerous in cruise. Owing to budget constraint on SLS installation, the operator (Uber) needs to place as few SLSs as possible and meets the quality of service (QoS) constraint. Safety is not optional but mandatory.

Optimization model goes into detail concerning the evolutionary pathway to a mass market through affordable vehicles and operations. The operational cost is mainly routing cost of VTOLs. It is our optimization goal to minimize it.

Multi-commodity flow problem is a network flow problem with multiple commodities (flow demands) between different source and sink nodes. The classification of flows is based on the number of paths to route one demand. Each VTOL takes a path and has a capacity on the number of passengers. Unsplittable multi-commodity flow problem [BHV00, LLRR20, Gen19] and k-splittable multi-commodity flow problem [GJPP10, TD08, TDM05] play a basic role in the derivation of VTOL traffic model. Unsplittable multi-commodity flow problem is proved to be $NP-$hard by Kleinberg et al. [Kle96, Kle98], k-splittable multi-commodity flow problem is proved to be $NP-$hard by Baier et al. [BKS02]. Multi-commodity flow problem [Alv05] can be reprensented by edge and path formulations. Formulations have significant impact on the speed of solver [Van00]. We develop two traffic models based on $k-$splittable and unsplittable multi-commodity flow models, and for each model edge and path formulations are proposed.

Both path formulations contain exponential number of path variables but have less constraints. We solve their LP relaxation by column generation approach. Column generation method [LD05] proved its efficiency to solve several multi-commodity flow and network optimization problems. To enforce the integrality, we use the branch-and-price algorithm [BJN$^+$98, GL14] based on the column generation method.

The remainder of this thesis is organized as follows. In Section (2), we introduce the problem description and notations of the transportation network, safety landing site (SLS), skyports, k-splittable and unsplittable flow models. In Section (3), we present edge and path formulations for k-splittable multi-commodity flow model. In Section (4), we present edge and path formulations for unsplittable multi-commodity flow model. In Section (5), we use a branch-and-bound algorithm for solving edge formulations. In Section (6), we introduce a branch-and-price algorithm for path formulations, the reduced cost and Farkas pricing and branching rules. In Section (7), we illustrate our instance generator and perform tests on various formulations and algorithms. Finally, we conclude this paper in Section (8) with prospect of research.

2 Problem Description

The model of vehicule transportation (taxi on ground) and its corresponding optimization problems are well studied. The model of VTOL transportation is still open for researchers, and we import basic concepts from network optimization to

model the VTOL transportation problem mathematically. We introduce networks of air traffic, flow representation of trajectories of VTOLs and constraints for placement of SLSs. These models play base roles to derive mathematical programs which are solvable by MILP solvers.

2.1 Air transportation and networks

The trajectory of the VTOL could pass through any part of the 3D continuous space, but usually the 3D space is sampled (discretized) to obtain a graph $G = (N, A)$. For all $(i, j) \in A$, let c_{ij} be the cost of arc (i, j).

As an example, let us consider Figure (1) which represents a simplified 2D problem, i.e., we neglect the altitude dimension for sake of clarity. The continuous space is sampled considering a regular grid and the set of nodes N is represented by the blue circles. The arcs A are, in this case, connecting each node i to its neighbours (the nodes within a distance to i equal to the sampling step). In the figure, each segment between two adjacent nodes represents two opposite direction arcs, but we explicitly draw the direction only for the path from s to t (in red). Note that, from a practical viewpoint, the identified path is smoothed in a second phase to make the trajectory more realistic. Of course, one could consider a different sampling and different arcs, like diagonal arcs. However, this topic is not covered in this work and remains an open question.

In case of congestion of VTOls, a capacity constraint m_{ij} is imposed on every edge $(i, j) \in A$, and it defines the maximum number of VTOLs. The weight c_{ij} is the cost of transportation per unit on that edge, and it is proportional to the length of the edge. These parameters are part of given data to represent congestion and cost, and we formulate them in the terminology of network optimization.

2.2 Traffic flow model

Skyports are subset of critical nodes in the networks for take off and landing. Transportation demands require to move passengers from one skyport to another. There are D demands in total with different sources and destinations.

Each demand h requires transportation of d_h from a source skyport s_h to a destination skyport t_h where d_h is the number of VTOLs.

2.2.1 k-splittable multi-commodity flows

In a scenario, the operator can use at most k paths to route one demand, therefore, corresponding VTOL takes one of these paths. Every path is unsplittable, and the routing problem is called k-splittable flow problem.

The flow on each path defines the number of VTOLs used. If the flow is integer, it is too hard for optimization. As paths are always the crucial results for the operator, we relax the integrality constraint and outputs the paths of flows, but fractional flow is a drawback of this model, for it cannot provide the accurate assignment of VTOLs to paths.

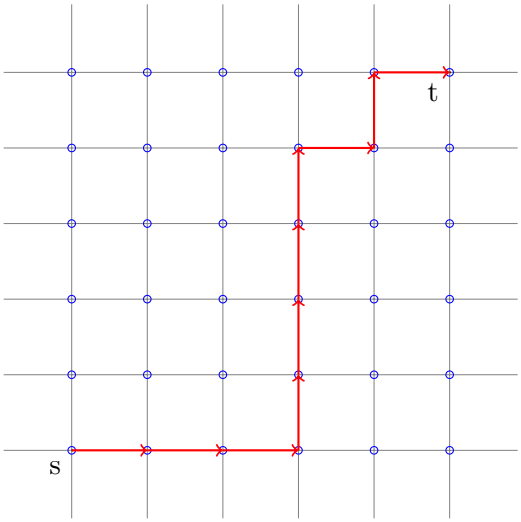

Figure 1: A path from skyport s to skyport t (in red)

2.2.2 Unsplittable multi-commodity flows

In the other scenario, the operator allows to choose the optimal path for every VTOL.

In fact, after assignment of passengers, demand h is decomposed into d_h single VTOL demands, every VTOL is equivalent to a demand, and the demand just defines the source and the destination of this VTOL. We view that a demand is transported by a VTOL or flow. Each flow now is unspittable, and the problem is called unsplittable multi-commodity flow problem. Flows of this problem is purely binary.

2.3 SLSs and covers

In this article, we consider an additional constraint, i.e., the safety measure that imposes the VTOLs to be "not too far" from a so-called Safety Landing Site (SLS) all along its trajectory from source s to destination t. If we focus on the problem of identifying the optimal trajectory as discussed above, given a set of SLS, we can impose the safety constraint by removing from the graph $G = (V, A)$ each arc (i, j) which do not satisfy it, i.e., for which there exists a point in the segment $i - j$ such that the distance between that point and the closest SLS is greater than the safety distance.

Usually, all SLSs have an identical covering range. For each SLS ℓ, its cover is a subset A_ℓ of edges of the network that are located in the safety distance from the SLS. SLSs are not necessarily vertices of the network, in fact, we define SLSs by their covering edges which are part of data for our algorithm. In particular, skyports are special SLSs although their cover range are smaller than that of SLS.

We denote the set of edges covered by skyports by A_0 for which SLSs are redundant. Therefore, VTOL have mutiple choices of SLSs to land according to which edge its trajectory takes.

There are ℓ possible locations for placement of SLSs. However, to route all demands, only a subset of SLSs is enough, and the operator asks the optimizer to save the investment on SLSs. Our goal is to design the placement of SLSs such that edges of trajectories of all VTOLs are covered by at least one of installed SLSs, meanwhile the cost of transportation is minimized and the number of installed SLSs does not exceed a budget b.

3 Formulations for k-splittable flows

We propose two formulations of the k-splittable flow model: the edge formulation and the path formulation. The path formulation is the Dantzig-Wolfe-Decomposition of the edge formulation where path variables are extreme points of the convex hull of flows of edge formulation.

Their linear relaxations have the same value, but sizes of LPs are not the same, and thus solving time of corresponding MILPs are different. In following subsections we describe these two mathematical formulations of the k-splittable flow model.

3.1 Edge formulation

The edge formulation is a three-index model. Flows are decomposed into edge variables indexed by the edge index, the demand index and the order of splittable flow. There are two kinds of edge variables x and f, x is the binary route variable and f is the fractional flow variable. Let us recall parameters

Parameters:

c_{ij}: The cost per VTOL of transport each edge (i, j).

m_{ij}: The capacity of each edge (i, j).

d_h: The traffic demand from the source s_h to the target t_h.

β_{ij}^h: The upperbound for a flow of demand h on the edge (i, j), that is $\min\{m_{ij}, d_h\}$.

b: the budget on the number of SLSs.

Besides edge variables, we also have a demand allocation variable w_q^h to assign the amount of the demand h to q-th splittable flow. Moreover, the placement of SLSs is denoted by y.

Variables:

f_{ijq}^h : The fractional flow variable indicating how much amount of the demand h is routed on edge (i, j) by the q-th splittable flow.

x_{ijq}^h : The binary flow variable indicating whether the demand h is routed on edge (i, j) by the q-th splittable flow.

w_q^h : The amount of demand h routed by the q-th splittable flow.

y_ℓ : The binary SLS variable indicating whether the SLS is installed at the ℓ-th site.

$$\min z = \sum_{(i,j)\in A}\sum_{h=1}^{D}\sum_{q=1}^{k} c_{ij} f_{ijq}^h \tag{1a}$$

$$f_{ijq}^h \le \beta_{ij}^h x_{ijq}^h \qquad \forall h=[1,D], \forall q=[1,k], \ \forall (i,j)\in A, \tag{1b}$$

$$\sum_{j\in V} x_{ijq}^h - \sum_{j\in V} x_{jiq}^h = 0 \qquad \forall h=[1,D], \forall q=[1,k], \ \forall i \in V\setminus\{s_h,t_h\}, \tag{1c}$$

$$\sum_{j\in V} x_{s_h jq}^h - \sum_{j\in V} x_{j s_h q}^h = 1 \qquad \forall h=[1,D], \forall q=[1,k], \tag{1d}$$

$$\sum_{j\in V} x_{j t_h q}^h - \sum_{j\in V} x_{t_h jq}^h = -1 \qquad \forall h=[1,D], \forall q=[1,k], \tag{1e}$$

$$\sum_{j\in V} f_{ijq}^h - \sum_{j\in V} f_{jiq}^h = 0 \qquad \forall h=[1,D], \forall q=[1,k], \ \forall i \in V\setminus\{s_h,t_h\}, \tag{1f}$$

$$\sum_{j\in V} f_{s_h jq}^h - \sum_{j\in V} f_{j s_h q}^h = w_q^h \qquad \forall h=[1,D], \forall q=[1,k], \tag{1g}$$

$$\sum_{j\in V} f_{j t_h q}^h - \sum_{j\in V} f_{t_h jq}^h = -w_q^h \qquad \forall h=[1,D], \forall q=[1,k], \tag{1h}$$

$$\sum_{q=1}^{k}\sum_{h=1}^{D} f_{ijq}^h \le m_{ij} \qquad \forall (i,j)\in A, \tag{1i}$$

$$\sum_{h=1}^{D} w_q^h = d_h \qquad \forall h=[1,k] \tag{1j}$$

$$x_{ijq}^h \le \sum_{\ell=1,(i,j)\in A_\ell}^{\ell} y_\ell \qquad \forall h=[1,D], \forall q=[1,k], \forall (i,j)\in A\setminus A_0, \tag{1k}$$

$$\sum_{\ell=1}^{\ell} y_\ell \le b \tag{1l}$$

$$f_{ijq}^h \in [0,\beta_{ij}^h] \qquad \forall h=[1,D], \forall q=[1,k], \forall (i,j)\in A, \tag{1m}$$

$$x_{ijq}^h \in \{0,1\} \qquad \forall h=[1,D], \forall q=[1,k], \forall (i,j)\in A, \tag{1n}$$

$$y_\ell \in \{0,1\} \qquad \forall \ell=[1,\ell] \tag{1o}$$

$$w_q^h \in [0,d_h] \qquad \forall h=[1,D], \forall q=[1,k] \tag{1p}$$

Link route constraint (1b): This constraint links the flow variable f and route variable x.

Route flow conservation constraint (1c) to (1e) : The route flow x is balanced at each vertex.

Fractional flow conservation constraint (1f) to (1h) : The fractional flow f is balanced at each vertex.

Edge capacity constraint (1i): The sum of all flows f routed over a edge does not exceed its capacity.

Demand distribution constraint (1j): The sum of demands routed by different splittable flows is equal to that demand.

Cover constraint (1k): For every route of each demand, there is at least one SLS that covers its edges.

Budget constraint (1l): The sum of SLSs cannot exceed the budget.

3.2 Path formulation

Path formulation reformulates the edge formulation by transforming edge variables into path variables. Path variables are incident vectors of edge variables. Notice that the order of splittable flow is presented in the edge formulation. However, path variables only have two indices, the path index and the demand index.

Parameters:

P^h: The set of simple paths from the source s_h to the destination t_h for demand h.

β^h: The upperbound for the flow of a path of demand h, that is $\min\{\max_{ij\in A} m_{ij}, d_h\}$.

Variables:

f_p^h : the fractional flow variable indicating how much amount of the demand h is routed on path p.

x_p^h : the binary flow variable indicating whether the demand h is routed on path p.

y_ℓ : the binary SLS variable indicating whether the SLS is installed.

$$\min z = \sum_{(i,j)\in A} c_{ij} \sum_{h=1}^{D} \sum_{p\in P^h,(i,j)\in p} f_p^h \tag{2a}$$

$$f_p^h \le \beta^h x_p^h \qquad \forall h = [1,D], \forall p \in P^h, \tag{2b}$$

$$\sum_{h=1}^{D} \sum_{p\in P^h,(i,j)\in p} f_p^h \le m_{ij} \qquad \forall (i,j) \in A, \tag{2c}$$

$$\sum_{p\in P^h} x_p^h \le k \qquad \forall h = [1,D], \tag{2d}$$

$$\sum_{p\in P^h} f_p^h = d_h \qquad \forall h = [1,D], \tag{2e}$$

$$\frac{1}{k} \sum_{p\in P^h,(i,j)\in p} x_p^h \le \sum_{\ell=1,(i,j)\in A_\ell}^{\ell} y_\ell \qquad \forall h = [1,D], \forall (i,j) \in A \setminus A_0, \tag{2f}$$

$$\sum_{\ell=1}^{\ell} y_\ell \le b \tag{2g}$$

$$f_p^h \in [0, \beta^h] \qquad \forall h = [1, D], \forall p \in P^h, \tag{2h}$$
$$x_p^h \in \{0, 1\} \qquad \forall h = [1, D], \forall p \in P^h, \tag{2i}$$
$$y_\ell \in \{0, 1\} \qquad \forall \ell = [1, \ell] \tag{2j}$$

Link route constraint (2b): This constraint links the flow path variable f and route path variable x.

Edge capacity constraint (2c): The sum of all flows f routed over a edge does not exceed its capacity.

Splittable constraint (2d): The number of flows are at most k.

Demand satisfication constraint (2e): The sum of flow path variables should equal to the demand.

Cover constraint (2f): For every path of each demand, there is at least some SLSs that cover its edges.

Budget constraint (2g): The sum of SLSs cannot exceed the budget.

Compared to the edge formulation, a large number of flow conservation constraints are no longer needed, but at the cost of exponential number (in the size of edges) path variables.

MILP solver relaxes above MIP model to LP, and this relaxation provides a lower bound of MIP useful for fathoming. After linearization, we have $f_p^h = \beta^h x_p^h$, and we can obtain a LP only with fractional x variables. The relaxation follows as

$$\min z = \sum_{(i,j)\in A} c_{ij} \sum_{h=1}^{D} \sum_{p\in P^h,(i,j)\in p} \beta^h x_p^h \tag{3a}$$
$$\sum_{h=1}^{D} \sum_{p\in P^h,(i,j)\in p} \beta^h x_p^h \le m_{ij} \qquad \forall (i,j) \in A, \tag{3b}$$
$$\sum_{p\in P^h} x_p^h \le k \qquad \forall h = [1, D], \tag{3c}$$
$$\sum_{p\in P^h} \beta^h x_p^h = d_h \qquad \forall h = [1, D], \tag{3d}$$
$$\frac{1}{k} \sum_{p\in P^h,(i,j)\in p} x_p^h \le \sum_{\ell=1,(i,j)\in A_\ell}^{\ell} y_\ell \qquad \forall h = [1, D], \forall (i,j) \in A \setminus A_0, \tag{3e}$$
$$\sum_{\ell=1}^{\ell} y_\ell \le b \tag{3f}$$
$$x_p^h \in [0, 1] \qquad \forall h = [1, D], \forall p \in P^h, \tag{3g}$$
$$y_\ell \in [0, 1] \qquad \forall \ell = [1, \ell] \tag{3h}$$

4 Formulations for unsplittable flows

We present an edge and a path formulations of unsplittable model in the same way as those of k-splittable model.

4.1 Edge formulation

In the edge formulation, it is a two-index model. Flows are decomposed into edge variables indexed by the edge index and the demand index. There are only one kind of edge variables x. Let us redefine some parameters

Parameters:

c_{ij}: The cost per unit (VTOL) of transport each edge (i, j).

m_{ij}: The capacity of each edge (i, j).

b: the budget on the number of SLSs.

Variables:

x_{ij}^h : The binary flow variable indicating whether the demand h is routed on edge (i, j).

y_ℓ : The binary SLS variable indicating whether the SLS is installed at the ℓ-th site.

$$\min z = \sum_{(i,j)\in A} \sum_{h=1}^{D} c_{ij} x_{ij}^h \tag{4a}$$

$$\sum_{j\in V} x_{ij}^h - \sum_{j\in V} x_{ji}^h = 0 \qquad \forall h = [1, D],\ \ \forall i \in V \setminus \{s_h, t_h\}, \tag{4b}$$

$$\sum_{j\in V} x_{s_h j}^h - \sum_{j\in V} x_{j s_h}^h = 1 \qquad \forall h = [1, D], \tag{4c}$$

$$\sum_{j\in V} x_{j t_h}^h - \sum_{j\in V} x_{t_h j}^h = -1 \qquad \forall h = [1, D], \tag{4d}$$

$$\sum_{h=1}^{D} x_{ij}^h \le m_{ij} \qquad \forall (i, j) \in A, \tag{4e}$$

$$x_{ij}^h \le \sum_{\ell=1,(i,j)\in A_\ell}^{\ell} y_\ell \qquad \forall h = [1, D], \forall (i, j) \in A \setminus A_0, \tag{4f}$$

$$\sum_{\ell=1}^{\ell} y_\ell \le b \tag{4g}$$

$$x_{ij}^h \in \{0, 1\} \qquad \forall h = [1, D], \forall (i, j) \in A, \tag{4h}$$

$$y_\ell \in \{0, 1\} \qquad \forall \ell = [1, \ell] \tag{4i}$$

Flow conservation constraint (4b) to (4d) : The route flow x is balanced at each vertex.

Edge capacity constraint (4e): The sum of all flows x routed over a edge does not exceed its capacity.

Cover constraint (4f): For every route of each demand, there is at least one SLS that covers its edges.

Budget constraint (4g): The sum of SLSs cannot exceed the budget.

4.2 Path formulation

The path formulation only contains binary flow variables and SLS variables. Let us define variables

Variables:

x_p^h : the binary flow variable indicating whether the demand h is routed on path p.

y_ℓ : the binary SLS variable indicating whether the SLS is installed.

$$\min z = \sum_{(i,j)\in A} c_{ij} \sum_{h=1}^{D} \sum_{p\in P^h,(i,j)\in p} x_p^h \tag{5a}$$

$$\sum_{h=1}^{D} \sum_{p\in P^h,(i,j)\in p} x_p^h \le m_{ij} \qquad \forall (i,j) \in A, \tag{5b}$$

$$\sum_{p\in P^h} x_p^h = 1 \qquad \forall h = [1, D] \tag{5c}$$

$$\sum_{p\in P^h,(i,j)\in p} x_p^h \le \sum_{\ell=1,(i,j)\in A_\ell}^{\ell} y_\ell \qquad \forall h = [1, D], \forall (i,j) \in A \setminus A_0, \tag{5d}$$

$$\sum_{\ell=1}^{\ell} y_\ell \le b \tag{5e}$$

$$x_p^h \in \{0, 1\} \qquad \forall h = [1, D], \forall p \in P^h, \tag{5f}$$

$$y_\ell \in \{0, 1\} \qquad \forall \ell = [1, \ell] \tag{5g}$$

Edge capacity constraint (5b): The sum of all flows x routed over a edge does not exceed its capacity.

Unsplittable constraint (5c): Demand is routed by an unsplittable flow.

Cover constraint (5d): For every path of each demand, there is at least some SLSs that cover its edges.

Budget constraint (5e): The sum of SLSs cannot exceed the budget.

4.3 Comparison

The columns and rows of edge formulations grow linearly with the size of networks, they are not scalable. Path formulations have many advantages, it contains less constraints. Although it consists of exponential number of path variables, it

can be solved efficiently by column generation. By adding the pricing algorithm into branch-and-bound procedure, we obtain the branch-and-price method.

5 Branch-and-bound

Branch-and-bound is an algorithm for solving discrete optimization problems, and it is widely used in MILP solvers. Branch-and-bound algorithm consists of a implicit enumeration of candidate solutions in the state space. The search tree partitions the state space, and its node represents a subset of the state space. At the start, the root node is the entire state space to explore. The search is guided by the LP relaxation of the problem. If the relaxed lower bound (dual bound) is greater than the incumbent value, the node is fathomed.If an inetger solution is found, the incumbent solution (the best solution found so far) is updated, or if the LP relaxation is infeasible, the node is aborted. Otherwise, an integer variable x has fractional value f, and the node is branched into two nodes. The up branch imposes the bound $x \leq \lfloor f \rfloor$, and the down branch imposes the bound $x \geq \lceil f \rceil$. Afterwards, a node is selected to repeat the above process until the duality gap (incumbent value minus the lowest dual bound) is zero.

Edge formulations k-splittable model (1) and unsplittable model (4) can be solved by branch-and-bound, but path formulation contains exponential number of path variables. It is impossible to list all path variables and solve the huge LP by branch-and-bound method, unless for tiny unrealistic instance.

6 Branch-and-Price

Some LP models consist of a huge number of variables compared to the number of constraints. The nonzero variables in optimal solution are at most as many as the number of constraints. Therefore, it is still possible to solve LP efficiently by recognizing the subset of variables which are non-zero in the optimal solution. The algortihm can start to solve a small subset of variables of initial LPs, and mange columns of LP dynamically. Column generation [LD05] is an efficient method for solving large scale linear programming, and a branch-and-bound method is called the branch-and-price when embedding it.

Usually, master problems of (3) and (5) have an exponential number of path variables, restricted master problem (RMP) limits columns of LP to a subset of path variables. At each iteration, the column generation method exploits the primal and dual information of the restricted master problem (RMP) to add new path variables with negative reduced cost into the LP.

Pricing algorithm generates new path variables which can improve the LP. The pricing algorithm consists of two parts: improve the current LP solution and repair the infeasibility. The first one is called reduced cost pricing, and it observes the reduced cost of path variables to improve the optimality. The second one is called Farkas pricing which identifies and repairs the infeasibility. We will construct

an extended graph in which the pricing algorithm is solved by the shortest path algorithm.

Both reduced cost pricing and Farkas pricing are derived from the dual problem or Lagrangian which defines dual constraints on path. In the next subsections, we will present the pricing problem for RMP of k-splittable model and unsplittable model.

6.1 Column generation for k-splittable flows

The linear relaxation of integer program of (2) relaxes integrity constraints on x and y. The following lemma guarantees that we can drop the upperbound on x and y in its LP relaxation.

Lemma 6.1. *If the problem (2) is feasible, then its LP relaxation admits an optimal solution with all $x \leq 1$ and $y \leq 1$.*

Proof. Since the problem (2) is feasible, there exist an optimal solution (x^*, f^*, y^*). If there exist a ${y^*}_p^h > 1$, by constraint (2c), ${x^*}_p^h \leq \beta_p^h$ always holds, decrease ${y^*}_p^h$ to 1, the solution remains feasible and optimal. If there exist a $y_\ell^* > 1$, because the left hand side of the constraint (2f) is always less or equal to 1, set y_ℓ^* to 1, the solution remains feasible and optimal. □

The bounds of variables of LP relaxation of (2) all becomes non-negative. We list the Lagrangian multipliers associated to constraints. The sign of the multiplier is defined by the sign of the constraints. If the constraint is in left hand side, the multiplier is non-negative. If the constraint is a equality constraint, the multiplier is unbounded. These multipliers appear to be dual variables of dual problem.

Dual variables by constraints:
Link route constraint (2b): $\alpha_p^h \geq 0, \forall h = [1, D], \forall p \in P^h$.
Edge capacity constraint (2c): $\gamma_{ij} \geq 0, \forall (i, j) \in A$.
Splittable constraint (2d): $\mu_h \geq 0, \forall h = [1, D]$.
Demand satisfication constraint (2e): $\zeta_h \in \mathbb{R}, \forall h = [1, D]$.
Cover constraint (2f): $\eta_{ij}^h \geq 0, \forall h = [1, D], \forall (i, j) \in A \setminus A_0$.
Budget constraint (2g): $\xi \geq 0$.

The Lagrangian on the LP relaxation defined as follow

$$
\begin{aligned}
L = & \sum_{(i,j)\in A} c_{ij} \sum_{h=1}^{D} \sum_{p\in P^h,(i,j)\in p} f_p^h + \sum_{h=1}^{D} \sum_{p\in P^h} \alpha_p^h (f_p^h - \beta^h x_p^h) + \sum_{\forall (i,j)\in A} \gamma_{ij} \\
& (\sum_{h=1}^{D} \sum_{p\in P^h,(i,j)\in p} f_p^h - m_{ij}) + \sum_{h=1}^{D} \mu_h (\sum_{p\in P^h} x_p^h - k) + \sum_{h=1}^{D} \zeta_h (\sum_{p\in P^h} f_p^h - \\
& d_h) + \sum_{h=1}^{D} \sum_{(i,j)\in A\setminus A_0} \eta_{ij}^h (\frac{1}{k} \sum_{p\in P^h,(i,j)\in p} x_p^h - \sum_{\ell=1,(i,j)\in A_\ell}^{\ell} y_\ell) + \\
& \xi (\sum_{\ell=1}^{\ell} y_\ell - b)
\end{aligned}
\tag{6}
$$

It is easy to check that LP relaxation of (2) equals to the primal of Lagrangian relaxation defined by the follow

$$
primal_1 = \min_{x\geq 0, f\geq 0, y\geq 0} \max_{\alpha\geq 0,\gamma\geq 0,\mu_h\geq 0,\zeta\in\mathbb{R},\geq,\eta\geq 0,\xi\geq 0} L. \tag{7}
$$

Thus we can define the dual of the LP relaxation of (2) by

$$
dual_1 = \max_{\alpha\geq 0,\gamma\geq 0,\mu\geq 0,\zeta\in\mathbb{R},\geq,\eta\geq 0,\xi\geq 0} \min_{x\geq 0, f\geq 0, y\geq 0} L. \tag{8}
$$

By the strong duality theorem of linear program

Theorem 6.1. *If either the primal or the dual have non-empty feasible set, then* $primal_1 = dual_1$.

Constraints of dual problem are based on primal variables, we deduce the dual as follow

$$
\max z = -\sum_{(i,j)\in A} \gamma_{ij} m_{ij} - \sum_{h=1}^{D} (\mu_h k + \zeta_h d_h) - \xi \sum_{\ell=1}^{\ell} b \tag{9a}
$$

$$
\sum_{(i,j)\in p} (c_{ij} + \gamma_{ij}) + \zeta_h + \alpha_p^h \geq 0, \qquad \forall h = [1, D], \forall p \in P^h \tag{9b}
$$

$$
-\sum_{h=1}^{D} \sum_{(i,j)\in A_\ell} \eta_{ij}^h + \xi \geq 0, \qquad \forall \ell = [1, \ell] \tag{9c}
$$

$$
\sum_{(i,j)\in p,(i,j)\notin A_0} \frac{\eta_{ij}^h}{k} + \mu_h - \beta^h \alpha_p^h \geq 0, \qquad \forall h = [1, D], \forall p \in P^h \tag{9d}
$$

$$
\alpha_p^h \geq 0, \qquad \forall h = [1, D], \forall p \in P^h \tag{9e}
$$

$$\gamma_{ij} \geq 0, \qquad \forall (i,j) \in A \tag{9f}$$
$$\mu_h \geq 0, \qquad \forall h = [1, D] \tag{9g}$$
$$\zeta_h \in \mathbb{R}, \qquad \forall h = [1, D] \tag{9h}$$
$$\eta_{ij}^h \geq 0, \qquad \forall h = [1, D], \forall (i,j) \in A \setminus A_0 \tag{9i}$$
$$\xi \geq 0. \tag{9j}$$

It is easy to verify that zero dual variables is a feasible solution to the dual problem. The dual problem have exponential number of constraints on path. Because the dual problem is non-empty, so we can use the dual simplex to solve the dual problem.

We can add constraints (9d) and (9b) of the dual problem with negative feasibility, in turn, those path variables x and f associated with these constraints are added into active columns of the primal problem. Accoding to the strong duality theorem and the complementary slackness, we can deduce the primal solution by the dual solution.

However, the number of dual varibales α is exponential, we cannot generate all α of the dual in advance. Notice that $f_p^h = \beta^h x_p^h$, so they are generated in pair. In the other way, we can derive the dual of the equivalent LP relaxation (3) in the following

$$\max z = -\sum_{(i,j)\in A} \gamma_{ij} m_{ij} - \sum_{h=1}^{D} (\mu_h k + \zeta_h d_h) - \sum_{h=1}^{D} \sum_{(i,j)\in A\setminus A_0} \eta_{ij}^h \sum_{\ell=1,(i,j)\in A_\ell}^{\ell} y_\ell - \xi \sum_{\ell=1}^{\ell} b \tag{10a}$$
$$-\sum_{h=1}^{D} \sum_{(i,j)\in A_\ell} \eta_{ij}^h + \xi \geq 0, \quad \forall \ell = [1, \ell] \tag{10b}$$
$$\beta^h \Big(\sum_{(i,j)\in p} (c_{ij} + \gamma_{ij}) + \zeta_h \Big) + \sum_{(i,j)\in p, (i,j)\notin A_0} \frac{\eta_{ij}^h}{k} + \mu_h \geq 0, \quad \forall h = [1, D], \forall p \in P^h \tag{10c}$$
$$\gamma_{ij} \geq 0, \quad \forall (i,j) \in A \tag{10d}$$
$$\mu_h \geq 0, \quad \forall h = [1, D] \tag{10e}$$
$$\zeta_h \in \mathbb{R}, \quad \forall h = [1, D] \tag{10f}$$
$$\eta_{ij}^h \geq 0, \quad \forall h = [1, D], \forall (i,j) \in A \setminus A_0 \tag{10g}$$
$$\xi \geq 0. \tag{10h}$$

This dual problem do not have dual variables associated with exponential number of path variables. We can solve it by improve and check procedure such that after the restricted dual problem is solved to optimal, we adds the most violated constraint to the dual and corresponding primal path variable into the primal.

The left hand side of dual constraints (10) associated to path variable $p \in P^h, h \in D$ is indeed the reduced cost of this path variable in the primal problem. Let us denote it as

$$RC(p) = \beta^h(\sum_{(i,j)\in p} (c_{ij} + \gamma_{ij}) + \zeta_h) + \sum_{(i,j)\in p,(i,j)\notin A_0} \frac{\eta_{ij}^h}{k} + \mu_h \tag{11}$$

The reduced cost is the steepest descent direction. Denote the objective of solution (x, y) by $z(x, y)$, we have the well known interpretation of reduced cost from the following lemma:

Lemma 6.2. *Let (x, y) be a basic feasible solution of (3). If increase x_p from 0 to arbitrarily small ϵ such that corresponding (x, y) remains feasible, then $RC(p) = \frac{z(x,y)z(x,y)}{\epsilon}$.*

Therefore, once there exists negative reduced cost, we can improve the primal by increase the associated column from zero, which is called column generation.

The part of reduced cost w.r.t the edge i.e. path cost follows as

$$PC(p) = \beta^h \sum_{(i,j)\in p} (c_{ij} + \gamma_{ij}) + \sum_{(i,j)\in p,(i,j)\notin A_0} \frac{\eta_{ij}^h}{k} \tag{12}$$

To find the least reduced cost path p, we can generate a graph identical to the network and add the cost c, γ and η to the weight of the edge. The shortest path in the graph is the least reduced cost path. Since all costs are non-negative, we can run the Dijkstra's algorithm whose complexity is $O(|V|^2)$. To check the optimality of the primal, we check whether the reduced cost of the path is negative, if it non-negative, the LP optimal, otherwise, we add the column and reoptimize.

Because we actually work on the LP relaxation of (2), and duals are obtained from its constriaints, then we add x_p and f_p associated to a shortest path p in the (2), and we should add the link constraint (2b) as well. The solver SCIP [GAB+20] supports to add columns during the pricing, but it cannnot support add rows in pricing shown in the Figure (2). Therefore, we cannot implement it by hand.

In this section, one can learn a trick to derive the reduced cost without writing the dual problem explicitly. We can just write the dual variables whose associated primal constraints contain path variables, for they compose the term w.r.t the path variables in the Lagrangian. Gathering all dual terms (dual variables times with some constants) associated with the same path, we obtain its reduced cost.

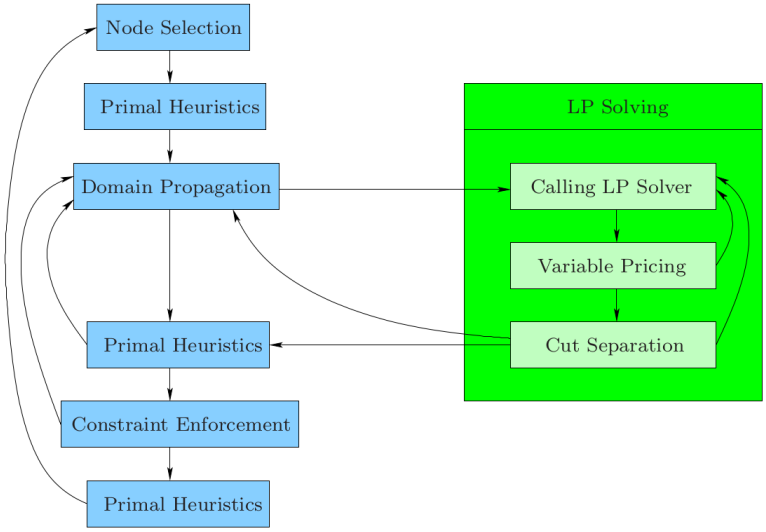

Figure 2: The framework of SCIP

6.2 Column generation for unsplittable flows

The continuous linear relaxation of integer program of (5) relaxes integrity constraints on x and y. Following the lemma (6.1) , we can drop the upperbound of y. The upperbound of x is implicitly implied by the constraint (5c), so we can drop the upperbound as well

Dual variables by constraints:

Edge capacity constraint (5b): $\gamma_{ij} \geq 0, \forall (i,j) \in A$.

Unsplittable constraint (5c): $\mu_h \in \mathbb{R}, \forall h = [1, D]$.

Cover constraint (5d): $\eta_{ij}^h \geq 0, \forall h = [1, D], \forall (i,j) \in A \setminus A_0$.

Budget constraint (5e): $\xi \geq 0$.

6.2.1 Reduced cost pricing

By the trick of the precedent section, we can deduce the reduced cost of $p \in P^h, h \in [1, D]$ as the follow

$$RC(p) = \sum_{(i,j)\in p} (c_{ij} + \gamma_{ij}) + \sum_{(i,j)\in p:(i,j)\notin A_0} \eta_{ij}^h + \mu_h \tag{13}$$

It is the left hand side of the dual constraints associated with the path variable p. Again, we use the Dijkstra's algorithm to compute the least reduced cost path, add the path if its reduced cost is negative. This approach is called reduced cost pricing. The reduced cost pricing problem observes whether there are path variables of negative reduced costs to improve the optimality. If RMP is solved to optimality

and then no variables have negative reduced costs, we can deduce the optimality of the master problem.

6.2.2 Farkas pricing

We initialize the RMP with the shortest path of each demand. If the primal of (5) is infeasible, so its dual is unbounded. During branch-and-price, it is possible when the presolver fails to find a feasible solution or the subproblem is infeasible after branching. The Farkas pricing [And01] identifies and repairs the infeasibility.

Let us recall a variant of Farka's lemma [CCZ+14]

Lemma 6.3. *The system* $Ax + By \leq f, Cx + Dy = g, x \geq 0, y \geq 0$ *is infeasible if and only if there exists* (u, v) *s.t.* $A^T u + C^T v \geq 0, B^T + D^v, u \geq 0$

Note that the Farkas condition $A^T u + C^T v \geq 0, B^T + D^v = 0, u \geq 0$ is exactly the tangent cone of dual constraints, where (u, v) is called Farkas certificate. According to Farkas' lemma, a RMP is infeasible if and only if there is an unbounded ray (Farkas certificate) of the dual problem of RMP. Since the branch-and-price algorithm is usually solved by MILP solver with dual-simplex method, it is easy to incorporate infeasibility detection into the solver.

The pricing algorithm detects the dual constraints (13) associated with the path variable for which the left hand side is negatively unbounded for the Farkas certificate. Adding this constraint would add a cutting plan to repair unboundness of the dual RMP and hence the infeasibility of the primal RMP.

If such Farkas certificate of the dual problem exists, the Farkas pricing adds the corresponding path variables into the RMP of the primal. Here, the Farkas coefficient $FC(p)$ of path variable x_p is computed by the same way as reduced cost except that edge cost $c = 0$ as follow

$$FC(p) = \sum_{(i,j)\in p} \gamma_{ij} + \sum_{(i,j)\in p, (i,j)\notin A_0} \eta_{ij}^h + \mu_h$$

If Farkas coefficients are non-negative for all paths, the primal problem is infeasible. Thus, the solver can abort this node. If Farkas coefficient of at least one path variable is negative, the solver adds it to RMP and reoptimize.

6.3 Branching rule

At every node in the branch-and-bound tree, if no more columns are priced out (LP is optimal) and the LP solution is still fractional, branching rules are called to enforce the integrity of path variables and SLS variables.

The branching rule for the problem (5) is decomposed into two phases, the first phase enforces the integrity of path variables and the second phase enforces the integrity of SLS variables. If there are fractional path variables, the algorithm goes to the first phase, otherwise, it goes to the second phase.

As for path variables, branching on variable is a straight way to fix the binary path variable x to either 0 or 1. If the variable is fixed to 1, there is no issue. If

the variable is fixed to 0, the pricing algorithm might regenerate it. In the worst case, the solver generates a column, fix it to zero and regenerate it again, and thus never terminates.

Another approach proposed in [BHV00] branches on edges. If a path of demand st is fractional, there must exist two fractional paths of flow st that diverge at two edges (i, j) and (i, k) from a common vertex i. We choose the demand with the most number of non-zero fractional flows. Then we partition the out edges from i into two sets such that partition of divergent edges are almost of the same size.

Therefore, for one branch, it can forbid the edge (i, j) by the constraint $\sum_{p \in \varphi^{st}:(i,j) \in p} x_p = 0$; for the other branch, it can forbid the edge (i, k) by the constraint $\sum_{p \in \varphi^{st}:(i,k) \in p} x_p = 0$.

Actually, we are adding forbidden edge set into two branches. Our branching rule follows the same methodology as [BHV00], but we emphasize more on dichtonomy and balance of search tree. We choose the demand h with maximum fractional paths, and create two disjoint edge sets such that partition of divergent edges are almost of the same size. We also try to maximize the number of edges to branch unlike only two branch edges. The algorithm follows as (1).

The pricing algorithm is compatible with branching decisions, it just deletes corresponding forbidden edges for each flow when computing the shortest path.

As for the SLS variable, we branch on SLS variable y_{ℓ^*} according to the most fractional rule [AKM05] as follows

$$\ell^* = \arg\max_{\ell} \min\{y_\ell, 1 - y_\ell\} \tag{14}$$

In the up branch, y_{ℓ^*} is fixed to 1; In the down branch, y_{ℓ^*} is fixed to 0. If the right hand side of the constraint (5d) in the down branch is zero, the corresponding

edges are forbidden.

Algorithm 1: Branching rule

Data:
G: the original graph representing network, $G = (V, E)$
P^h: the set of non-zero path variables of demand $h \in D$
Result: Forbidden edges
$h \longleftarrow \arg\max_h |P^h|$
Find the first divergent node i for paths in P^h
$O_p^h \longleftarrow$ out edges from i of P^h
$O_G^h \longleftarrow$ non-forbidden out edges from i in G
$E_1^h, E_2^h \longleftarrow \emptyset$
for $(i, j) \in O_p^h$ **do**
  if $|E_1^h| < |O_2^h|$ **then**
    $E_1^h \longleftarrow E_1^h \cup \{(i, j)\}$
  else
    $E_2^h \longleftarrow E_2^h \cup \{(i, j)\}$
for $(i, j) \in O_G^h \setminus O_p^h$ **do**
  if $|E_1^h| < |O_2^h|$ **then**
    $E_1^h \longleftarrow E_1^h \cup \{(i, j)\}$
  else
    $E_2^h \longleftarrow E_2^h \cup \{(i, j)\}$
return E_1^h, E_2^h

7 Experiments

As there is lack of public data set for SLS locations, we generated networks, skyports, SLSs and traffic demands artificially. The design of instance generator follows the guidelines of the Uber team and whitepaper [RZS18]. With these instances, we test the performance of edge formulation (1) for $k-$splittable flow models, edge and path formulations (4 and (5)) for unsplittable models.

7.1 Instance generator

We generate for each test instance, a meshed network with SLSs, skyports, demands, capacities and costs. Conceptually, the map is a $[0, 1] \times [0, 1]$ rectangle map. The network is represented by a mesh of this map. If the mesh has $L_x \times L_y$ nodes, the mesh has the length L_x and the width L_y. SLSs are circles in the map. Skyports are selected as parts of mesh nodes. Demands are represented by (s_h, t_h, d_h), the source and target nodes and the demand value. We partition the map into cells, placement of SLSs and skyport are cells by cells. The scale ratio defines the width over length, and cell and mesh units in y and x axis follow this ratio.

Inputs of the instance generator follow as:

1. sr: the scale ratio of y axis of over x axis of the map.
2. L_x: the number of nodes in a row along x-axis.
3. S_x: the number of cells for skyport localization in a row along x-axis,
4. ℓ_x: the number cells for SLS localization in a row along x-axis,
5. D: the number of demands in the unit of VTOLs,
6. $rSLS$**:** the scale factor for SLSs's ellipse.
7. rfS**:** the scale factor for skports.
8. ℓ_{cell}: the number SLSs in each cell,
9. C: the set of capacities in the unit of VTOLs.

Outputs of the instance generator follow as:

1. $G = (V, A)$: the mesh network with costs and capacities assigned to edges,
2. D demands with source, target and demand values,
3. A_0: the cover set by skyports A_0.
   A_ℓ: other cover set by SLSs $\forall \ell \in \ell$.

Dallas Metropolitan is a template to launch the VTOL service, we assume the length and width of this city is about 100km and 100km. The VTOL has landing range of 20km (150mph in 5 mins), the size range is 96 km. Therefore, there are 25 SLSs, for each the the covered radius is 20 km. The number of skyports are 64. By inputting these parameter into the instance generator, we can generate a network with the same scale like Dallas in Figure 4

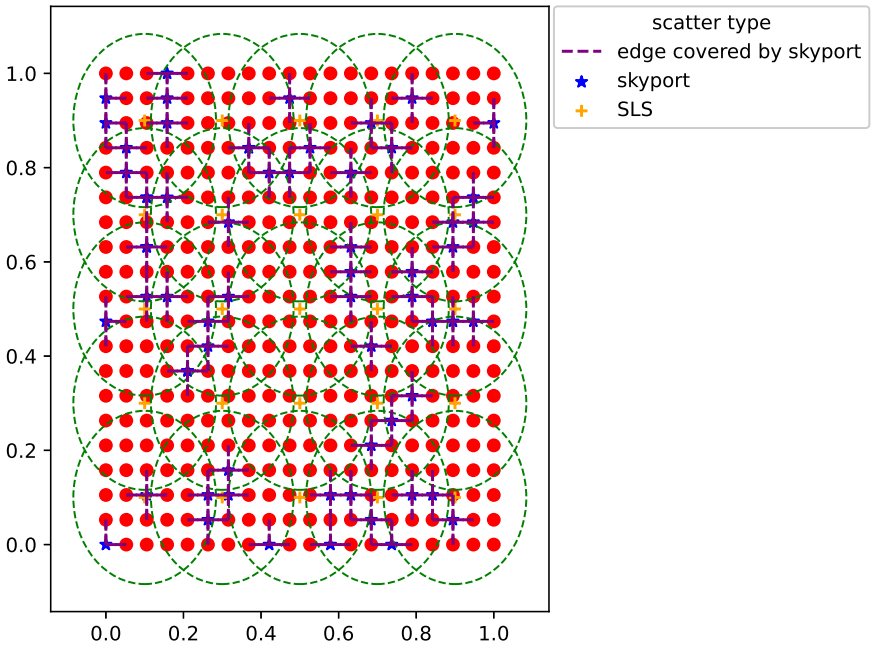

Figure 3: A network like Dallas Metropolitan

The network in Figure (4) is generated with the setting of $sr = 1.2, L_x = 6, S_x = 2, \ell_x = 4, D = 7, rfSLS = 1.6, rfS = 0.5, C = \{1, 2, 3\}$ such that $A_0 = \emptyset$. This regular network is easy to adjust its scale, and we use them to test the scalability of algorithms in subsequent experiments.

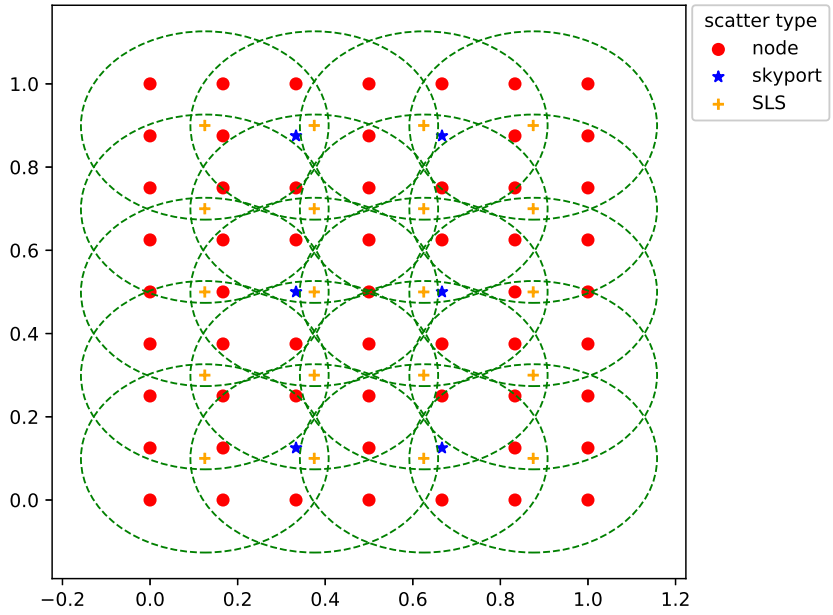

Figure 4: A network

The budget limits the number of SLSs available which in turn contrained routing paths of VTOLs, and the optimal solutions would be different under different budget constraint. For the instance in Figure (4), we set a budget of 5 and 8 respectively. Optimal solutions are depicted in Figure (5) and (6). Black ellipses are installed SLSs. On the budget of 5, the routing cost is 591.19; On the budget of 8, the routing cost is 472.00. More budgets provides more routing possibilities, so the cost is lower.

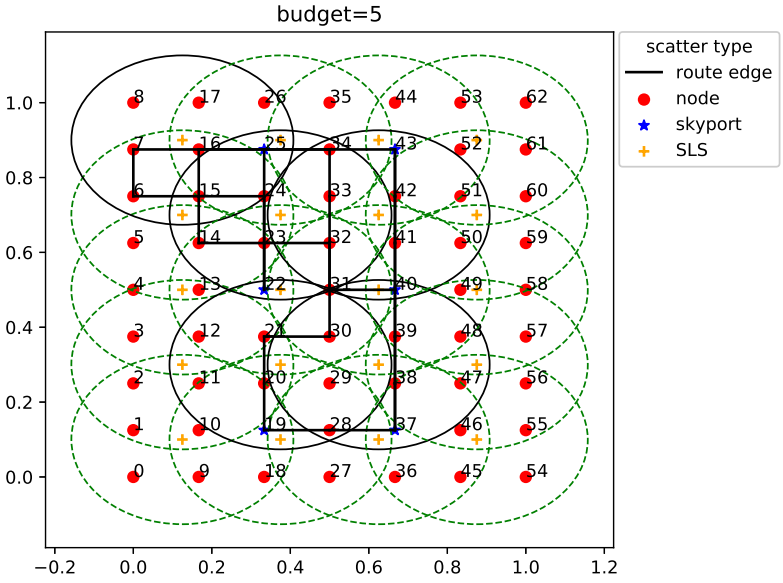

Figure 5: Budget 5

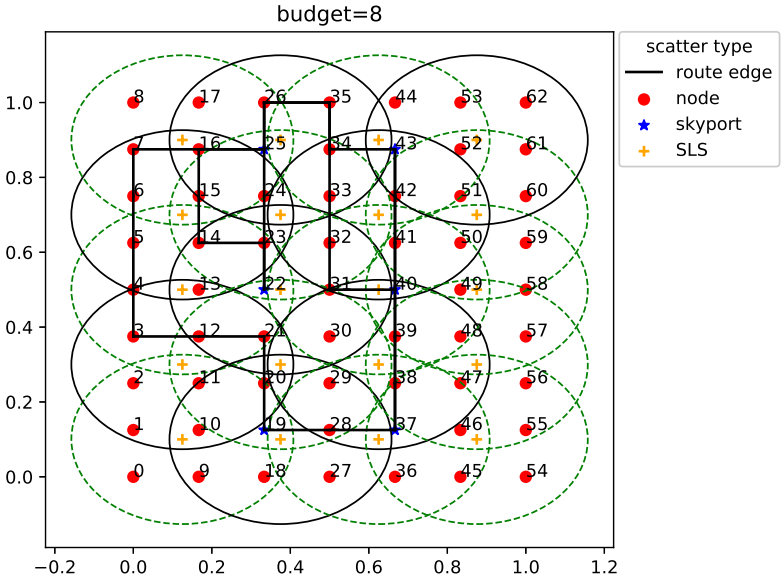

Figure 6: Budget 8

7.2 Numerical comparison of algorithms

In this section, we test performance of models and algorithms. Tested algorithms are the performance of the branch-and-price algorithm for path formulation (5) of unsplittable flow model, the branch-and-bound algorithm for edge formulation (4) of unsplittable flow model and the branch-and-bound algorithm for edge formulation (1) of k-splittable (k=3) flow model.

The computing environment has an Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz and 16GB of RAM with Ubuntu 20.04 operating system. We implement branch-and-price algorithm by MILP solver SCIP [GAB+20] with IBM ILOG CPLEX 12.10.0 as LP solver, branch-and-bound algorithm by the MILP solver CPLEX 12.10.0 in single-thread mode. Time limit is set to 3600 seconds.

Data set consists of 8 instances, and the number of nodes ranges from 36 to 529. All networks are grid meshes, Table (1) lists numbers of nodes, edges, demands, SLSs and skyports, respectively.

Instance	Nodes	Edges	Demands	SLSs	Skyports
1	36	120	3	16	4
2	48	164	6	20	6
3	63	220	7	20	6
4	100	360	11	36	16
5	225	840	17	49	36
6	324	1224	20	64	49
7	400	1520	25	81	64
8	529	2024	25	100	81

Table 1: Instance

In our experiments, we fix a budget for each instance. We test the branch-and-bound algorithm for edge formulation of unsplittable flow problem (4), the branch-and-price algorithm for path formulation of unsplittable flow problem (5) and the branch-and-bound algorithm for edge formulation of 3-splittable flow problem (1). For each algorithm on instances, we list the primal bound (incumbent solution) $\overline{z}^*$, duality gap (%), time (seconds) and the number of nodes. The symbol '$-$' indicates that the solver does not find a solution in the time limit.

We first compare the performance of formulation (4) and (1), as their models are different but solved by the branch-and-bound method of the same solver Cplex. For formulation (4), 6 instances (4) are solved to optimality, 1 instance has a feasible solution and 1 instance has no feasible solution found; For formulation (1), 4 instances (4) are solved to optimality and 4 instances have no feasible solution found. As for providing a feasible solution, (4) is always faster, as it has less variables and constraints than (1). However, as for instances with feasible solution for formulation (1), the optimal objective is always lower. In the unsplittable model,

I	B	Edge formulation unsplittable				Path formulation unsplittable				Edge formulation k-splittable			
		$\overline{z}^*$	Gap	t	Nodes	$\overline{z}^*$	Gap	t	Nodes	$\overline{z}^*$	Gap	t	Nodes
1	5	175.98	0	0.02	1	175.98	0	0.53	4	162.25	0	0.54	231
2	5	355.92	0	0.05	1	355.92	0	0.2	23	348.97	0	10.99	1344
3	5	591.19	0	4.74	1538	591.19	0	3600	128920	456.65	0	9.27	622
4	5	300.05	0	0.05	1	300.05	0	0.56	1	298.01	0	0.71	1
5	9	1512.13	0	22.83	1446	1512.18	0.31	3600	64666	-	-	3600	777
6	20	2290.75	0	790.37	20861	-	-	3600	33192	-	-	3600	616
7	25	3025.70	0.35	3600	30341	-	-	3600	10635	-	-	3600	225
8	29	-	-	3600	20861	-	-	3600	10829	-	-	3600	1

Table 2: Performance experiments

every demand is exactly one VTOL, which is routed by a single path, so the flow is always binary. In the 3-splittable model, paths of demand might have fractional VTOLs. If these demands are routed by unsplittable flows, then paths must have integer VTOLs. Therefore, the 3-splittable model is more flexible than unsplittable model and it relaxes the integrality of its fractional flows f. To refine the solution of (1), the optimizer can start to solve the model of large k, enforce the integerality constraint on f, and reoptimize the path formulation on the restricted path set from the previous result.

Then, we compare the performance of formulation (4) and (5), they solve the same model by different algorithms. As for the branch-and-price algorithm for (5), it only solves 1 instance to optimality, it finds feasible solutions for 4 solutions and it does not find solutions for 3 instances. It is not stable and slower than edge formulation (4). If its initial path variables are close to the optimal solution, it can prove the optimality very fast like in instance 4. It spends lots of time proving the optimality of instance 3 although it already finds the optimal solution. It is a common problem in column generation that the dual problem converge slowly, some stabilization techniques [PSUV13, PSUV18] can accelerate the convergence.

8 Conclusions

In this work we proposed mathematical programming approaches for minimizing energy cost of VTOLs in traffic networks under safety and budget constraints.

We derived new path formulations for unsplittable and $k-$splittable flow models respectively. We developed a column generation approach embedded in a branch-and-price algorithm.

In the future, we can develop a polyhedral study of our problem to devise new valid inequalities as well as a branch-price-cut (BPC) algorithm. The widely used inequalities [AR02, BBA07, BHV00, VA18] for unsplittable multi-commodity flow problem deals with knapsack capacity constraint, and we can import it in our problem.

According to the order of branching rules, we actually solve master and sub

problem iteratively. The master problem first fixes the binary SLS variables, and the sub problem searches for the binary flow variables. We can apply Bender decomposition [Cos05] into the branch-and-price algorithm, when the sub problem is infeasible or non-optimal, the so-called no-good cuts [CX04] can strengthen the original integer program.

To model the uncertainty of demands, we can use the robust linear programming models, which the program would be converted to a second order cone programming problem. Its main problem is efficiency of second order cone solver.

9 Acknowledge

The author would like to thank Claudia D'Ambrosio, Leo Liberti and Sonia Vanier for their helpful comments and advice in the internship.

10 Summary of the Internship

Regeneration of columns for k-splittable model: LP (3) is equivalent to LP relaxation of (2), they only differ in the linking constraint (2b), therefore, duals of other identical constraints should be the same as well. To avoid the issue of unknown dual in (2), the reduced cost (11) is derived from (3).

Notice: the reduced cost (11) is manually derived from duals, not internal reduced cost provided by scip (In a correct implementation, they should be the same!).

The MIP program given to scip is (2). The scip cannot support to add rows (2b) during column generation for LP relaxation of (2). The scip's internal LP is different is from ideal LP (2) and (3) that I supposed in my paper. Therefore, the duals that I have are wrong, for sure, the reduced cost is wrong. The reduced cost of a generated column becomes negative (would be regenerated!) is sometimes -70, so it is not numerical rounding error.

Stability: As Leo pointed, column generation has a big problem in stablility. The problem is another issue different from regeneration of columns. In this case, the convergence is very slow, it is observed in my 'correct' implementation of unsplittable model (5), the dual bound does not change for very long time, even it already finds the best solution (primal bound) same as that of edge formulation implemented in cplex.

Bibliography

[AKM05] Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

[Alv05] Filipe Pereira Alvelos. Branch-and-price and multicommodity flows. 2005.

[And01] Erling D Andersen. Certificates of primal or dual infeasibility in linear programming. *Computational Optimization and Applications*, 20(2):171–183, 2001.

[AR02] Alper Atamtürk and Deepak Rajan. On splittable and unsplittable flow capacitated network design arc–set polyhedra. *Mathematical Programming*, 92(2):315–333, 2002.

[BBA07] Meriema Belaidouni and Walid Ben-Ameur. On the minimum cost multiple-source unsplittable flow problem. *RAIRO-Operations Research*, 41(3):253–273, 2007.

[BHV00] Cynthia Barnhart, Christopher A Hane, and Pamela H Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2):318–326, 2000.

[BJN+98] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[BKS02] Georg Baier, Ekkehard Köhler, and Martin Skutella. On the k-splittable flow problem. In *European Symposium on Algorithms*, pages 101–113. Springer, 2002.

[CCZ+14] Michele Conforti, Gérard Cornuéjols, Giacomo Zambelli, et al. *Integer programming*, volume 271. Springer, 2014.

[Cos05] Alysson M Costa. A survey on benders decomposition applied to fixed-charge network design problems. *Computers & operations research*, 32(6):1429–1450, 2005.

[CX04] Yingyi Chu and Quanshi Xia. Generating benders cuts for a general class of integer programming problems. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pages 127–141. Springer, 2004.

[GAB+20] Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, et al. The scip optimization suite 7.0. 2020.

[Gen19] Bernard Gendron. Revisiting lagrangian relaxation for network design. *Discrete Applied Mathematics*, 261:203–218, 2019.

[GJPP10] Mette Gamst, Peter Neergaard Jensen, David Pisinger, and Christian Plum. Two-and three-index formulations of the minimum cost multicommodity k-splittable flow problem. *European Journal of Operational Research*, 202(1):82–89, 2010.

[GL14] Bernard Gendron and Mathieu Larose. Branch-and-price-and-cut for large-scale multicommodity capacitated fixed-charge network design. *EURO Journal on Computational Optimization*, 2(1-2):55–75, 2014.

[HG16] Jeff Holden and Nikhil Goel. Fast-forwarding to a future of on-demand urban air transportation. *San Francisco, CA*, 2016.

[Kle96] Jon M Kleinberg. Single-source unsplittable flow. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 68–77. IEEE, 1996.

[Kle98] Jon M Kleinberg. Decision algorithms for unsplittable flow and the half-disjoint paths problem. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 530–539, 1998.

[LD05] Marco E Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations research*, 53(6):1007–1023, 2005.

[LLRR20] Markus Leitner, Ivana Ljubić, Martin Riedler, and Mario Ruthmair. Exact approaches for the directed network design problem with relays. *Omega*, 91:102005, 2020.

[PSUV13] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and Francois Vanderbeck. In-out separation and column generation stabilization by dual price smoothing. In *International Symposium on Experimental Algorithms*, pages 354–365. Springer, 2013.

[PSUV18] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, 30(2):339–360, 2018.

[RZS18] Hamed Rahimi, Ali Zibaeenejad, and Ali Akbar Safavi. A novel iot architecture based on 5g-iot and next generation technologies. In *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 81–88. IEEE, 2018.

[TD08] Jérôme Truffot and Christophe Duhamel. A branch and price algorithm for the k-splittable maximum flow problem. *Discrete Optimization*, 5(3):629–646, 2008.

[TDM05] Jérôme Truffot, Christophe Duhamel, and Philippe Mahey. Using branch-and-price to solve multicommodity k-splittable flow problems. In *Proceedings of International Network Optimization Conference (INOC)*. Citeseer, 2005.

[VA18] Sonia Vanier and Knippel Arnaud. New partition inequalities for the unsplittable flow problem. In *ISCO International Symposium on Combinatorial Optimization*, 2018.

[Van00] François Vanderbeck. On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48(1):111–128, 2000.