Exact Solution of Hub Network Design Problems with Profits

Armaghan Alibeyg[a], Ivan Contreras[a], Elena Fernández[b,c]

[a] *Concordia University and Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada H3G 1M8*
[b] *Statistics and Operations Research Department, Universitat Politècnica de Catalunya, Barcelona, Spain*
[c] *Barcelona Graduate School of Mathematics*

Abstract

This paper considers hub network design problems with profits in which the simultaneous optimization of the collected revenue, fixed setup cost of the hub network and variable transportation cost are considered. An exact algorithmic framework is proposed for two variants of this class of problems, where a sophisticated Lagrangean function that exploits the structure of the problems is used to efficiently obtain bounds at the nodes of an enumeration tree. In addition, reduction tests and partial enumerations are used to considerably reduce the size of the problems and thus help decrease the computational effort. Numerical results on a set of benchmark instances with up to 100 nodes confirm the efficiency of the proposed algorithmic framework.

Keywords: location; hub network design; hub location; Lagrangean relaxation; branch-and-bound.

1. Introduction

Large-scale transportation and telecommunications networks arising in air and ground transportation, postal delivery, and rapid transit systems frequently use hub-and-spoke architectures to efficiently route flows. Transshipment, consolidation, or sorting points, referred to as hub facilities, are

Email addresses: a_alibey@encs.concordia.ca (Armaghan Alibeyg), icontrer@encs.concordia.ca (Ivan Contreras), e.fernandez@upc.edu (Elena Fernández)

employed in these networks to connect a large number of origin/destination (O/D) pairs indirectly by using a small number of links. *Hub location problems* (HLPs) consider the design of hub networks by selecting a set of nodes to locate hubs, activating a set of links, and routing a predetermined set of commodities through the network while optimizing a cost-based (or service-based) objective function.

This paper studies *hub network design problems with profits* (HNDPPs), a class of HLPs recently introduced in Alibeyg et al. (2016). HNDPPs release the classical requirement of most HLPs that all service demand must be satisfied, and incorporate one additional level to the decision making process so as to determine the O/D nodes and associated commodities whose demand must be served. The rationale behind HNDPPs is that in many applications a revenue is obtained for serving the demand of a given commodity. Capturing such a revenue is likely to incur not only a routing cost but also additional setup costs, as the O/D nodes of the served commodities may require the installation of additional infrastructure. Classical HLPs, however, ignore these considerations, as reflected by the requirement that the demand of every commodity must be served. Broadly speaking, this requirement expresses the implicit hypothesis that the overall cost of solution networks will be compensated by the overall revenue. Since such hypothesis does not necessarily hold, incorporating decisions on the nodes where service should be offered and the commodities that should be routed have important implications in the strategic and operational costs.

Potential transportation applications of HNDPPs arise in the airline and ground transportation industries. In the case of airline companies, network planners have to design their transportation network when they are first entering into the market, or may have to modify already established hub-and-spoke networks through alliances, merges and acquisitions of companies. The involved decisions are to determine the cities that will be part of their network, i.e. what cities they will provide service to (served nodes) and what O/D pairs (served commodities) will be served in order to offer air travel services to passengers (served demand) between city pairs. Additional decisions focus on the location of their main airports (hub facilities) and on the selection of the legs used for connecting regional airports (served nodes) with hub airports and for connecting some hub airports between them. Finally, the transportation of passengers using their established network. The objective is to find an optimal hub network structure that maximizes the total net profit for providing air travel services to a set O/D pairs while taking into account the (re)design cost of the network.

To the best of our knowledge the literature on HNDPPs reduces to Alibeyg et al. (2016), where several variants of HNLPPs are introduced and analyzed. These models incorporate into the decision-making process additional strategic decisions on the nodes and the commodities that will be served. They consider profit-oriented objectives that measure the tradeoff between the revenue due to served commodities and the overall network design and transportation costs. Broadly speaking the proposed HNDPPs are of three types: (i) primary profit-oriented models, which may or may not consider service commitment constraints, (ii) profit-oriented models with network design decisions that incur setup costs on the edges used on service routes, and (iii) more complex models with multiple demand levels and possibly multiple service levels as well. The results of extensive computational experiments reported in Alibeyg et al. (2016) illustrate the characteristics of the solution networks produced by these different models, as well as the computational difficulty for solving them with a state-of-the art commercial solver. In particular, the results also show that, despite the advantages that HNDPPs may bring to the decision maker, the proposed formulations are very demanding from a computational point of view, in terms of both computing time and memory when used with a commercial solver. For the primary profit oriented model without any additional service commitment constraints, instances with up to 70 nodes can be solved to optimality in one day of CPU time, and when such additional constraints are added only instances with up to 60 nodes can be solved. When approaching the more complex models with multiple demand and service levels, only instances with up to 35 nodes can be solved in the same time limit.

In this paper we focus on methodological aspects leading to the exact solution of the two primary HNDPPs presented in Alibeyg et al. (2016). The first one, denoted as PO_1, is flexible in the sense that among all commodities associated with served O/D nodes, only those that are actually profitable are routed. It is applicable in situations where there are no service commitments or external regulations imposing the decision maker to serve any commodity whose O/D nodes are both activated. The second model, denoted as PO_2, considers a more restrictive scenario in which such commitments or regulations do exist and thus, all commodities whose O/D nodes are both activated would have to be served, even if this would reduce the total profit.

The main contribution of this paper is to propose a unified algorithmic framework applicable to large-scale instances of both PO_1 and PO_2 models involving up to 100 nodes. It is an exact branch-and-bound procedure in which a sophisticated Lagrangean relaxation is used to obtain tight bounds at each node of the enumeration tree. In particular, the proposed Lagrangean function resorts to the solution of well-known *quadratic boolean problems* (QBPs). We show how, due to the special cost structure associated with the quadratic term of the objective function, the QBPs can be efficiently solved by transforming them to classical *minimum cut problems*. The algorithm is enhanced through several algorithmic refinements that make it more efficient. These include: (i) variable elimination techniques that allow reducing considerably the size of the formulations at the root node, (ii) a partial enumeration phase capable of effectively exploring the solution space by reducing the required number of nodes in the tree, and (iii) the use of simple but effective primal heuristics embedded in the subgradient algorithm that exploit the structure of the problem. Computational experiments confirm the effectiveness of our exact algorithmic framework since it is able to obtain optimal solutions for instances with up to 100 nodes for both PO_1 and PO_2, whereas a commercial solver can only handle instances with up to 70 and 60 nodes, respectively.

The remainder of the paper is organized as follows. Section 2 reviews relevant literature related to HNDPPs. In Section 3 we introduce the formal definition and mixed integer programming (MIP) formulations of PO_1 and PO_2. Section 4 describes the proposed Lagrangean relaxations of PO_1 and PO_2 and the solution of their associated Lagrangean duals. Section 5 explains the variable elimination techniques used whereas Section 6 presents the partial enumeration and the overall branch-and-bound algorithm. Section 7 describes the computational experiments we have run. Conclusions follow in Section 8.

2. Literature Review

HNDPPs extend *hub arc location problems* (HALPs) by selecting the nodes to be served and the commodities to be routed. That is, HNDPPs incorporate an additional level to usual HALP decisions. In its turn, HALPs extend fundamental HLPs (see, Campbell and O'Kelly, 2012; Contreras, 2015) by incorporating network design decisions dealing with the selection of the hub arcs that can be used in O/D paths, in addition to classical hub location and allocation decisions. Different HALPs have been studied in the literature. For instance, HALPs with a cardinality constraint on the number of opened hub arcs (Campbell et al., 2005), HALPs that incorporate setup costs for the hub nodes and hubs arcs (Contreras and Fernández, 2014; Gelareh et al., 2015), or HALPs that impose particular topological structures, such as tree-star (Contreras et al., 2010), star-star (Labbé and Yaman, 2008), ring-star (Contreras et al., 2016), and hub lines (Martins de Sá et al., 015a,b). We can also relate HNDPPs to studies that focus on the design of hub networks in airline transportation (see, for instance, Aykin, 1995; Jaillet et al., 1996; Sasaki et al., 1999; Bryan and O'Kelly, 1999; O'Kelly, 2012; Saberi and Mahmassani, 2013). We note that all these works focus on the location of hubs, but ignore other relevant decisions addressed in HNDPPs, like the nodes to be served and the commodities to be routed.

Contrary to most HLPs and HALPs that optimize a cost-based (or service-based) objective, HNDPPs deal with a profit-oriented objective which optimizes the profit expressed as the difference between the revenue obtained for the service offered and the costs due to the design of the network and to transportation. This feature relates HNDPPs to two families of HLPs, aiming at the maximization of the profit obtained for serving nodes and routing commodities: *maximal hub covering problems* (MHCPs), and *competitive hub location problems* (CHLPs). In MHCPs, demand is covered if both origin and destination nodes are within a specified distance of a hub node. These problems were introduced by Campbell (1994) and more recently extended by Hwang and Lee (2012) and Lowe and Sim (2012). Similarly to HNDPPs, MHCPs allow some commodities not to be served (in this case due to covering constraints). However, like in the previous HLPs mentioned above, MHCPs do not incorporate decisions on the nodes to be served, which are essential in HNDPPs.

From a different perspective, CHLPs focus on the design of hub networks within the framework of competing firms. Most CHLPs assume that a company already operates in the market (leader), and address the maximization of demand captured by a new company who wants to enter the market (follower). That is, the usual objective in CHLPs is to maximize the market share of the new firm. Marianov et al. (1999) introduce CHLPs with two competitors in which the follower looks for the best location for its hubs so as to maximize its captured demand, assuming the single allocation of customers to open hubs. The first proposed model assumes that if the routing cost of of the new firm does not exceed the current competitor's cost for a given O/D pair, then its associated flow will be fully captured. The second model is more flexible as it allows a fraction of the demand flow to be captured. This feature is modeled using a stepwise linear function, which is used for the comparison with the competitor's routing costs. In both models, at most one path can be used to route commodities between each O/D pair. Eiselt and Marianov (2009) extended that work by considering a gravity like attraction function that allows using more than one path for routing a given commodity. This utility is assumed to depend on three factors. The first factor is the basic attractiveness of a pair of hub in each trip by each airline company. The second factor is the cost of the route and the third is the total time required by a flight.

Lüer-Villagra and Marianov (2013) study a competitive model in which pricing decisions are involved and the aim is to maximize the profit of the entering company rather than its market share. The new company (entrant) looks for the best hub network and prices that maximizes his/her profit. The entrant's profit is defined as the difference between the net revenue minus the fixed and variable costs. Contrary to Eiselt and Marianov (2009), in which customer's choice are represented using a gravity model, Lüer-Villagra and Marianov (2013) represent the customers' behavior using a logit discrete choice model which reflects customers' sensitivity to prices. When the customers' sensitivity to prices is low, they do not necessarily look for the route with the cheapest price which has been considered by Marianov et al. (1999). The authors also confirm that in a competitive environment, maximizing the market share is dominated by profit maximization.

Other CHLPs have been studied, for instance, by Gelareh et al. (2010), who present a model arising in liner shipping networks, where a new liner service provider designs its network to maximize its market share, using a stepwise attraction function, which depends on service times and routing costs. and by Mahmutogullari and Kara (2016) who present hub-medianoid and hub-centroid CHLPs where the market is assumed to be a duopoly, customers select one firm based on the provided service levels and the objective is to maximize their market share. The interested reader is addressed to Adler and Smilowitz (2007), Lin and Lee (2010), and Sasaki et al. (2014), for examples of approaches of CHLPs under a game theoretic framework.

O'Kelly et al. (2015) study a hub location model with price-sensitive demands that considers three different service levels for routing commodities: two-hub O/D paths, one-hub O/D paths, and direct connections. The model is formulated as an economic equilibrium problem that maximizes a nonlinear concave utility function minus the sum of the setup and routing costs. One key difference between HNDPPs and CHLPs is the competitive framework in

CHPLs, which contrasts to the one single firm's framework of the HNDPPs. Another difference is that, to the best of our knowledge, none of the studied CHLPs deal with servicing decisions for O/D nodes.

3. Formal Definition and Formulation of the HNDPP

In this section we formally define the considered primary HNDPPs and provide their associated modeling assumptions. The reader is referred to Alibeyg et al. (2016) for an extensive analysis of these modeling assumptions and their implications.

Let $G = (N, A)$ be a directed graph, with $|N| = n$, and let also $H \subset N$ be the set of potential hub locations. We denote by $A_H = \{(i, j) \in A \mid i, j \in H\} \subset A$ the subset of arcs connecting two potential hub nodes, where it is possible that $i = j$. We also consider the set of edges connecting two potential hubs, denoted as $E_H = \{\{i, j\} \mid i, j \in H\}$. Any edge $\{i, j\} \in E_H$ is indistinctively denoted as $\{j, i\}$. The elements of E_H are called *hub edges*. In the literature hub edges are often referred to as hub arcs but, like in Alibeyg et al. (2016), we prefer to maintain the distinction between edges and arcs. Service demand is given by a set of commodities that we denote by K. Each $k \in K$ is defined as a triplet $(o(k), d(k), W_k)$, where $o(k), d(k) \in N$, respectively denote its origin and its destination, also referred to as its O/D pair, and $W_k \geq 0$ denotes its service demand, i.e., the amount of flow that must be routed from $o(k)$ to $d(k)$ if commodity k is served.

Each node $i \in N$ will be of exactly one of the following types: a hub node, a served node, or an unserved node. If a node $i \in N$ is activated either as hub or as served, then it will be possible to route commodities with origin or destination at i. On the contrary, no commodity originated or with destination at an unserved node can be routed. Each served node must be assigned to at least one hub node and we allow multiple assignments, i.e. the assignment need not be unique. These assignments will be used to define the paths that serve commodities starting or terminating at the served nodes.

For $(i, j) \in A$, $d_{ij} \geq 0$ denotes the unit transportation (routing) cost between nodes i and j, which we assume to be symmetric, i.e., $d_{ij} = d_{ji}$, and to satisfy the triangle inequality. Associated with each $i \in N$, $c_i \geq 0$ denotes the setup cost for serving node i. If a node $i \in H$ is selected to be a hub, a fixed setup cost $f_i \geq 0$ is incurred. In this case it will be possible to route commodities with origin or destination at $i \in H$ without incurring the *service* setup cost c_i. Edges in E_H can be activated incurring setup costs. We denote by $r_e \geq 0$ the setup cost of $e \in E_H$. Activating a hub edge also requires to activate its end-nodes as hub nodes, and allows sending flows through any of its associated arcs with discounted transportation costs. If $\{i, j\} \in E_H$ is activated, the per unit flow cost through its associated arcs $(i, j), (j, i) \in A_H$ is αd_{ij}, where the parameter $\alpha, (0 \leq \alpha \leq 1)$ is used as a discount factor.

In the HNDPP, the effect of serving (routing) commodity $k \in K$ is threefold. On the one hand, it forces the activation of its O/D nodes $o(k)$ and $d(k)$. On the other hand, it produces a per unit revenue $R_k \geq 0$, which is independent of the path used to send the commodity demand W_k through the solution network. Finally, serving commodity k also incurs a transportation cost, which depends on W_k and on the path that is used to route it from $o(k)$ to $d(k)$. Similarly to most HLPs, in the HNDPP all O/D paths used to route served commodities must include at least one hub node and at most three edges. Hence, solution networks contain no direct connections between two non-hub nodes. For a served commodity k, let $(o(k), i, j, d(k))$ denote the path connecting $o(k)$ and $d(k)$. In this path it is required that i (resp. j) be a hub to which $o(k)$ (resp. $d(k)$) is assigned. Moreover, when $i \neq j$ the intermediate leg, $\{i, j\}$, must be associated with a hub edge. O/D paths of the form $(o(k), o(k), d(k), d(k))$, using just one hub arc, may arise only when both $o(k)$ and $d(k)$ are hub nodes. O/D paths with $i = j$ do not use any hub arc and consist solely of the collection and distribution legs, i.e. $(o(k), i, i, d(k))$ (origin-hub-destination) with $o(k) \neq i$ and $d(k) \neq i$. The per unit transportation cost for routing commodity k via the path $(o(k), i, j, d(k))$ is defined as $F_{ijk} = (\chi d_{o(k)i} + \alpha d_{ij} + \delta d_{jd(k)})$, where the parameters χ and δ reflect weight factors for collection and distribution, respectively.

The HNDPP consists of: (i) selecting a set of nodes to be served, (ii) locating a set of hub facilities, (iii) activating a set of hub edges, (iv) selecting a set of commodities to be served, both of whose O/D nodes have been selected in (i) and, (v) determining the paths to route the selected commodities through the solution network, with the objective of maximizing the difference between the total revenue obtained for serving the demand of the selected commodities minus the sum of the setup costs for the design of the network and the transportation costs for routing the commodities.

We next provide an MIP formulation for the first primary HNDPP, denoted as PO_1, in which no service commitments are imposed. For $i \in H$, we introduce binary location variables z_i equal to 1 if and only if a hub is located at node i, and for $i \in N$ we define binary variables s_i equal to 1 if and only if node i is served (i.e. activated as a non-hub node). For $e \in E_H$, we define y_e equal to 1 if and only if hub edge e is activated. Finally, for $k \in K$, $i, j \in H$, let the routing variable x_{ijk} represent the fraction of commodity k routed via arc $(i,j) \in A_H$. When $i = j$, $x_{iik} = 1$ indicates that commodity k is routed through the path $(o(k), i, d(k))$ visiting only hub i and thus, does not use any hub edge. Using these sets of variables, Alibeyg et al. (2016) formulate the HNDPP as follows:

$$
\begin{aligned}
(PO_1) \quad \text{maximize} \quad & \sum_{k \in K} \sum_{(i,j) \in A_H} W_k(R_k - F_{ijk}) x_{ijk} - \sum_{i \in H} f_i z_i - \sum_{i \in N} c_i s_i \\
& - \sum_{e \in E_H} r_e y_e && && (1) \\
\text{subject to} \quad & s_i + z_i \leq 1 && i \in H && (2) \\
& \sum_{(i,j) \in A_H} x_{ijk} \leq s_{o(k)} + z_{o(k)} && k \in K && (3) \\
& \sum_{(i,j) \in A_H} x_{ijk} \leq s_{d(k)} + z_{d(k)} && k \in K && (4) \\
& \sum_{j \in H} x_{ijk} + \sum_{j \in H : i \neq j} x_{jik} \leq z_i && k \in K, i \in H && (5) \\
& x_{ijk} + x_{jik} \leq y_e && k \in K, e = \{i,j\} \in E_H && (6) \\
& x_{ijk} \geq 0 && k \in K, (i,j) \in A_H && (7) \\
& z_i \in \{0,1\} && i \in H && (8) \\
& s_i \in \{0,1\} && i \in N && (9) \\
& y_e \in \{0,1\} && e \in E_H. && (10)
\end{aligned}
$$

The first term of the objective function is the net profit of the commodities that are routed. The other terms represent the total setup costs of the hubs that are chosen, the non-hub nodes that are selected to be served, and the hub edges that are used. Constraints (2) guarantee that if a node is activated as a hub then it is not activated as a served node. Constraints (3) and (4) impose that the O/D nodes of each routed commodity are activated, either as hub or served nodes. When $o(k)$ or $d(k)$ do not belong to H then the right hand side of constraints (3) and (4) reduces to $s_{o(k)}$ and $s_{d(k)}$, respectively. Constraints (5) prevent commodities from being routed via non-hub nodes, whereas constraints (6) activate hub edges. Finally, constraints (7)–(10) define the domain for the decision variables. Finally, (1)–(10) uses

$|N| + |H| + |E_H|$ binary variables, $|K||A_H|$ continuous variables, and $|H| + |K|(2 + |H| + |E_H|)$ constraints.

Even though (1)–(10) does not consider integrality conditions of the routing variables x, it always exist an optimal solution of (1)–(10) in which all x variables are integral. This is a consequence of the lack of capacity restrictions on the amount of flow routed though the hub nodes and hub arcs. For a given commodity, it may happen that two or more paths have the same (minimum) transportation cost, which may cause the commodity to be routed via more than one O/D path. However, the solution can always be modified (without affecting the objective value) to use exactly one path for every routed commodity on the solution network.

An extension of the above primary HNDPP, denoted as PO_2, considers service commitments that impose to serve any commodity whose O/D nodes are both activated, even if this would reduce the total profit. An MIP formulation for this more restrictive model can be obtained by adding to (1) - (10) the following set of constraints (Alibeyg et al., 2016):

$$s_{o(k)} + z_{o(k)} + s_{d(k)} + z_{d(k)} \leq \sum_{(i,j)\in A_H} x_{ijk} + 1 \qquad k \in K. \tag{11}$$

Constraints (11) force to route any commodity where both its O/D nodes are activated. PO_2 has the same number of variables as PO_1 but $|K|$ additional constraints. The effect of constraints (11) in the actual difficulty for solving the problem is notorious. The results of Alibeyg et al. (2016) show that the required CPU times for solving PO_2 with a commercial solver are at least one order of magnitude higher than those of PO_1 for all considered benchmark instances. As we will show later in Section 7, our algorithmic framework is capable of considerably mitigating the effect of (11) in the CPU times.

4. Lagrangean Relaxation

Lagrangean relaxation (LR) is a well-known decomposition method that exploits the inherent structure of the problems to compute dual bounds on the value of the optimal solution. Pirkul and Schilling (1998), Elhedhli and Wu (2010), and Contreras et al. (011b) provide some examples of successful implementations of LR for obtaining tight bounds for various classes of HLPs.

Our algorithmic framework uses LR to obtain upper bounds of PO_1 and PO_2. In the case of PO_1 we relax the sets of constraints (5) and (6), whereas for PO_2 we also relax the additional set of constraints (11). Hence, the structure of the resulting Lagrangean function is very similar in both cases: the domain is the same and only the objective functions differ. In both cases the Lagrangean function can be decomposed in two subproblems: one of them is trivial and the other one can transformed into a QBP. Due to the structure of the cost coefficients, we show how the Lagrangean function can actually be evaluated in polynomial time. We next provide the details of the entire process for PO_1 and then briefly describe how to proceed in a similar fashion for PO_2.

4.1. The Lagrangean function for PO_1

When we relax constraints (5) and (6), and incorporate them to the objective function of PO_1, with weights given by a multiplier vector (λ, μ) of appropriate dimension, we obtain the following Lagrangean function:

$$\begin{aligned} L_1(\lambda,\mu) = \text{maximize} \quad & \sum_{k\in K}\sum_{(i,j)\in A_H} W_k(R_k - F_{ijk})x_{ijk} - \sum_{i\in H} f_i z_i - \sum_{i\in N} c_i s_i \\ & - \sum_{e\in E_H} r_e y_e - \sum_{k\in K}\sum_{i\in H}\lambda_{ik}\big(\sum_{j\in H} x_{ijk} + \sum_{j\in H: i\neq j} x_{jik} - z_i\big) \\ & - \sum_{e=\{i,j\}\in E_H}\sum_{k\in K}\mu_{ek}(x_{ijk} + x_{jik} - y_e) \\ \text{subject to} \quad & (2)-(4), (7)-(10), \end{aligned}$$

which is equivalent to

$$\begin{aligned} L_1(\lambda,\mu) = \text{maximize} \quad & \sum_{k\in K}\sum_{(i,j)\in A_H} \overline{P}_{ijk} x_{ijk} - \sum_{i\in H}\overline{f}_i z_i - \sum_{i\in N} c_i s_i - \sum_{e\in E_H}\overline{r}_e y_e \\ \text{subject to} \quad & (2)-(4), (7)-(10), \end{aligned}$$

where

- $\overline{P}_{ijk} = \begin{cases} (R_k - F_{ijk})W_k - \lambda_{ik} - \lambda_{jk} - \mu_{\{i,j\}k}, & \text{if } (i \neq j) \\ (R_k - F_{iik})W_k - \lambda_{ik}, & \text{if } (i = j), \end{cases}$
- $\overline{f}_i = f_i - \sum_{k\in K} \lambda_{ik}$,
- $\overline{r}_e = r_e - \sum_{k\in K} \mu_{ek}$.

Note that $L_1(\lambda, \mu)$ can be decomposed in two independent subproblems, one in the y space, that we denote $L_y(\mu)$, and another one in the (z, s, x) space, that we denote $L_{z,s,x}(\lambda, \mu)$. The first subproblem reduces to

$$L_y(\mu) = \max \left\{ -\sum_{e \in E_H} \overline{r}_e y_e : y \in \{0,1\}^{|E_H|} \right\},$$

and an optimal solution can be obtained by inspection. That is, we set $y_e = 1$ for all $e \in E_H$ with $\overline{r}_e < 0$, and $y_e = 0$ otherwise. Subproblem $L_{z,s,x}(\lambda, \mu)$ can be stated as

$$\begin{aligned} L_{z,s,x}(\lambda, \mu) = \text{maximize} \quad & \sum_{k \in K} \sum_{(i,j) \in A_H} \overline{P}_{ijk} x_{ijk} - \sum_{i \in H} \overline{f}_i z_i - \sum_{i \in N} \overline{c}_i s_i \\ \text{subject to} \quad & (2) - (4), (7) - (9). \end{aligned}$$

We next show that $L_{z,s,x}(\lambda, \mu)$ can be reformulated as a QBP involving only $|N|$ binary variables.

4.1.1. Solution to Subproblem $L_{z,s,x}(\lambda, \mu)$

Given (2), for each $i \in H$ we can replace $s_i + z_i$ with a new binary variable h_i, with cost coefficient $F_i = \min \left\{ c_i, \overline{f_i} \right\}$. For each $i \in N \setminus H$ we just define $h_i = s_i$ with coefficient $F_i = c_i$. We can now express $L_{z,s,x}(\lambda, \mu)$ as

$$\begin{aligned} L_{h,x}(\lambda, \mu) = \text{maximize} \quad & \sum_{k \in K} \sum_{(i,j) \in A_H} \overline{P}_{ijk} x_{ijk} - \sum_{i \in N} F_i h_i \\ \text{subject to} \quad & \sum_{(i,j) \in A_H} x_{ijk} \leq h_{o(k)} \qquad k \in K \qquad (12) \\ & \sum_{(i,j) \in A_H} x_{ijk} \leq h_{d(k)} \qquad k \in K \qquad (13) \\ & h_i \in \{0,1\} \qquad i \in N. \end{aligned}$$

Given that (12) and (13) imply that, in an optimal solution to $L_{h,x}(\lambda, \mu)$ when both $h_{o(k)} = h_{d(k)} = 1$, commodity k will be routed via arc $(i_k, j_k) \in \arg\max \left\{ \overline{P}_{ijk} : (i,j) \in A_H \right\}$, provided $\overline{P}_{i_k j_k k} > 0$. This allows us to project out the x_{ijk} variables and to rewrite $L_{h,x}(\lambda, \mu)$ only in terms of the h variables. For each $k \in K$, let $\overline{Q}_k = \max \left\{ 0, \max_{(i,j) \in A_H} \left\{ \overline{P}_{ijk} \right\} \right\}$ and

$$L_h(\lambda, \mu) = \max \left\{ \sum_{k \in K} \overline{Q}_k h_{o(k)} h_{d(k)} - \sum_{i \in N} F_i h_i : h \in \{0,1\}^{|N|} \right\}.$$

We note that the only difference between the above expression for $L_h(\lambda, \mu)$ and a standard QBP formulation is that the former is stated on a directed graph, whereas QBP is typically stated on an undirected graph. Indeed, this difference can be easily overcome by redefining the cost coefficients as follows. For each pair $l, m \in N$, with $l < m$ let $k, \bar{k} \in K$ denote the two commodities with endnodes l and m, i.e. $o(k) = l$, $d(k) = m$, and $o(\bar{k}) = m$, $d(\bar{k}) = l$. By setting $Q_{lm} = \overline{Q}_k + \overline{Q}_{\bar{k}}$, we finally obtain the following QBP reformulation of $L_{z,s,x}(\lambda, \mu)$:

$$L_h(\lambda, \mu) = \max \left\{ \sum_{l,m \in N : l < m} Q_{lm} h_l h_m - \sum_{i \in N} F_i h_i : h \in \{0, 1\}^{|N|} \right\}.$$

Although QBP is $\mathcal{NP}$-hard in the general case, there are some particular cases which are known to be polynomially solvable. Picard and Ratliff (1975) show that when all cost coefficients of the quadratic term are non-negative, the QBP reduces to a *minimum cut problem* in an auxiliary network. Given that by definition $Q_{lm} \geq 0$ for all $l, m \in N$, $l < m$, for any feasible multiplier vector $(\lambda, \mu) \geq 0$, $L_h(\lambda, \mu)$ can thus be evaluated in polynomial time. For the sake of completeness, we next provide a sketch of the procedure to define the auxiliary network used for solving $L_h(\lambda, \mu)$ as a minimum cut problem. The reader is addressed to Picard and Ratliff (1975) for further details.

Let $G^{Aux} = (V^{Aux}, A^{Aux})$ be a digraph where the set of nodes V^{Aux} contains the original nodes $l \in N$, denoted as v_l, plus an artificial source s_0 and an artificial sink s_n. The set of arcs A^{Aux} is characterized as follows. There is an arc (s_0, v_l) connecting the source with each $l \in N$ of capacity $\sum_{m \in N} Q_{lm}$, if $F_l \geq 0$, and $2(\sum_{m \in N} Q_{lm} - F_l)$, otherwise. There is also an arc (v_l, s_n) connecting each $l \in N$ with the sink of capacity F_l, if $F_l \geq 0$, and $\sum_{m \in N} Q_{lm} - F_l$, otherwise. For each pair $l, m \in N$ with $l < m$ there is also an arc (v_l, v_m) with capacity Q_{lm}. Finally, there is an arc (s_0, s_n) with capacity $K - \sum_{l,m \in N : l < m} Q_{lm}$, where $K = \sum_{l,m \in N : l < m} Q_{lm}$.

Any (s_0, s_n)-cut in the above network can be associated with a solution $\bar{h}$ to $L_h(\lambda, \mu)$ (and vice-versa) as follows. If, for a given $l \in N$, (s_0, v_l) does not belong to the (s_0, s_n)-cut, then $\bar{h}_l = 1$ in the associated solution to $L_h(\lambda, \mu)$. Moreover, the arcs of the cut of the form (v_l, v_m) correspond to the pairs $l, m \in N$, $l < n$, where both $\bar{h}_l = \bar{h}_m = 1$. Furthermore, the value of the cut is precisely the value of $L_h(\lambda, \mu)$ for the solution $\bar{h}$ plus the constant K. An optimal solution to $L_h(\lambda, \mu)$ can thus be obtained by finding a minimum (s_0, s_n)-cut in G^{Aux}.

An optimal solution $(\bar{z}, \bar{s}, \bar{x})$ to $L_{z,s,x}(\lambda, \mu)$ in the original space can be retrieved from an optimal solution $(\bar{h}, \bar{y})$ to $L_h(\lambda^t, \mu^t)$ as follows. Note first that the only non-zero components of $\bar{x}$ are associated with commodities $k \in K$ with $\bar{h}_k = 1$. For each such commodity, we set $\bar{x}_{i_k j_k k} = 1$ if $\overline{P}_{i_k j_k k} > 0$, and 0 otherwise. As for the s variables, we set $\bar{s}_i = \bar{h}_i$ for each $i \in N \setminus H$ such that $F_i = c_i$, and 0 otherwise. Finally, we set $\bar{z}_i = \bar{h}_i$ for all $i \in H$ such that $F_i = \overline{f_i}$, and 0 otherwise.

Proposition 1. *For a given vector of multipliers (λ, μ), the Lagrangean function $L_1(\lambda, \mu)$ can be solved in $O(|K||A_H| + |N|^3)$ time.*

Proof The solution of $L_y(\mu)$ has complexity $O(|E_H|)$, which is dominated by the evaluation of coefficients Q_{lm} for $l, m \in N$ for $l < m$, with complexity $O(|K||A_H|)$. Given that $|V^{Aux}| = O(|N|)$ and $|A^{Aux}| = O(|N|^2)$, the solution of $L_{z,s,x}(\lambda, \mu)$ can be obtained in $O(|N|^3)$ time using the max-flow algorithm given in Orlin (2012) and the result follows. ■

4.1.2. Solution to the Lagrangean Dual

In order to obtain the best upper bound for PO_1 using $L_1(\lambda, \mu)$ we solve its associated Lagrangean dual problem

$$(D_1) \qquad Z_{D_1} = \min_{(\lambda,\mu)\geq 0} \quad L_1(\lambda, \mu) = L_y(\mu) + L_h(\lambda, \mu).$$

We use subgradient optimization to solve D_1. The algorithm follows the usual iterative scheme $(\lambda^{t+1}, \mu^{t+1}) = (\lambda^t, \mu^t) + \varepsilon_t \gamma^t$, where ε_t is the step length and γ^t is a subgradient of L_1 at (λ^t, μ^t). A subgradient of L_1 at a given point (λ^t, μ^t) can be easily obtained from an optimal solution $(\bar{s}, \bar{z}, \bar{x}, \bar{y})$ to $L_1(\lambda^t, \mu^t)$. In particular,

$$\gamma^t = \left(\left(\sum_{j \in H} \bar{x}_{ijk} + \sum_{j \in H : i \neq j} \bar{x}_{jik} - \bar{z}_i \right)_{i,k}, (\bar{x}_{ijk} + \bar{x}_{jik} - \bar{y}_e)_{i,j,e} \right).$$

We update the step length according to $\varepsilon_t = \Lambda^t (L_1(\lambda^t, \mu^t) - \underline{\eta}) / ||\gamma^t||^2$, where $\underline{\eta}$ is a valid lower bound on the optimal value of PO_1 and Λ^t is a given parameter whose value is updated at certain iterations (see Section 7.1 for the specific details of our implementation). Algorithm (1) summarizes the subgradient optimization algorithm that we apply. The algorithm terminates when one of the following criteria is met: (i) all the components of the subgradient are zero. In this case the current solution is proven to be optimal, (ii) the difference between the upper and lower bounds is bellow a threshold value, i.e., $|Z_{D_1} - Z^*| < \epsilon$, (iii) there is no improvement on the value of the upper bound after *niter* consecutive iterations, and (iv) the maximum number of iteration $Iter_{max}$ is reached.

Algorithm 1 Subgradient Optimization for PO_1

```
**Initialization**
    $Z_{D_1} = +\infty$; Initialize $(\lambda^0, \mu^0)$; $\Lambda^0$
    Let $\underline{\eta}$ be a lower bound on the optimal solution value
**while** Stopping criteria not satisfied **do**
    Solve $L_1(\lambda^t, \mu^t)$ and obtain an optimal solution $(\bar{s}, \bar{z}, \bar{x}, \bar{y})$
    **if** $L_1(\lambda^t, \mu^t) < Z_{D_1}$ **then**
        $Z_{D_1} \leftarrow L_1(\lambda^t, \mu^t)$
    **end if**
    Compute the subgradient $\gamma^t$
    Compute the step length $\varepsilon_t \leftarrow \Lambda^t(L_1(\lambda^t, \mu^t) - \underline{\eta})/||\gamma^t||^2$
    $(\lambda^{t+1}, \mu^{t+1}) \leftarrow (\lambda^t, \mu^t) + \varepsilon_t \gamma^t$
    $t \leftarrow t + 1$
**end while**
```

4.2. The Lagrangean Function for PO_2

Similarly to PO_1, in our LR of PO_2 we relax (5) and (6), incorporating them to the objective function with a multiplier vector (λ, μ). Moreover, we also relax (11), weighted with a multiplier vector π. An important property of this relaxation is that the domain of the Lagrangean function

$$L_2(\lambda, \mu, \pi) = \sum_{k \in K} \pi_k + \max \quad \sum_{k \in K} \sum_{(i,j) \in A_H} \overline{P}_{ijk} x_{ijk} - \sum_{i \in H} \overline{f}_i z_i - \sum_{i \in N} \overline{c}_i s_i - \sum_{e \in E_H} \overline{r}_e y_e$$
$$\text{s.t.} \quad (2) - (4), (7) - (10),$$

where

- $\overline{P}_{ijk} = \begin{cases} (R_k - F_{ijk})W_k - \lambda_{ik} - \lambda_{jk} - \mu_{\{i,j\}k} - \pi_k, & \text{if } (i \neq j) \\ (R_k - F_{iik})W_k - \lambda_{ik} - \pi_k, & \text{if}(i = j), \end{cases}$

- $\overline{c}_i = c_i - \sum_{k \in K : o(k) = i \text{ or } d(k) = i} \pi_k,$

- $\overline{f}_i = f_i - \sum\limits_{k \in K} \lambda_{ik} - \sum\limits_{k \in K: o(k)=i \text{ or } d(k)=i} \pi_k$,
- $\overline{r}_e = r_e - \sum\limits_{k \in K} \mu_{ek}$,

remains the same as in $L_1(\lambda, \mu)$ and the only difference is the objective function. It now consists of the constant $\sum_{k \in K} \pi_k$, which does not appear in $L_1(\lambda, \mu)$ but is irrelevant for the optimization, and two terms, one in the y space, which has exactly the same cost coefficients as in $L_1(\lambda, \mu)$, and another one in the (z, s, x) space, where the cost coefficients are now different from those of $L_1(\lambda, \mu)$. As before, $L_{z,s,x}(\lambda, \mu, \pi)$ can be transformed into a QBP on an undirected graph with non-negative cost coefficients. Thus, $L_2(\lambda, \mu, \pi) = \sum_{k \in K} \pi_k + L_y(\mu) + L_{z,s,x}(\lambda, \mu, \pi)$ can also be solved in polynomial time by transforming $L_{z,s,x}(\lambda, \mu, \pi)$ into a min-cut problem.

Similarly to PO_1, in order to obtain the best upper bound for PO_2 using $L_2(\lambda, \mu, \pi)$ we solve its associated Lagrangean dual problem

$$(D_2) \qquad Z_{D_2} = \min_{(\lambda,\mu,\pi) \geq 0} \quad L_2(\lambda, \mu, \pi) = \sum_{k \in K} \pi_k + L_y(\mu) + L_{z,s,x}(\lambda, \mu, \pi).$$

We apply a subgradient optimization algorithm similar to Algorithm 1 for solving the Lagrangean dual. Details are omitted.

4.3. Lower Bounds from Primal Solutions

In this section we explain how feasible solutions are constructed to obtain valid lower bounds for PO_1 and PO_2. In particular, we exploit the information generated from the integer solutions to the Lagrangean duals at some iterations of the corresponding subgradient optimization algorithms.

4.3.1. A Primal Heuristic for PO_1

Let $(\bar{s}, \bar{z}, \bar{x}, \bar{y})$ denote the solution to $L_1(\lambda, \mu)$ at the current iteration. Since in $L_1(\lambda, \mu)$ the sets of constraints (5) and (6) are relaxed, the solution $(\bar{s}, \bar{z}, \bar{x}, \bar{y})$ may not be feasible for PO_1. We next describe a simple heuristic to obtain a feasible solution $(\hat{s}, \hat{z}, \hat{x}, \hat{y})$ to PO_1.

The initial solution is the outcome of $L_1(\lambda, \mu)$ but with all routing variables at value zero, i.e., initially, $(\hat{s}, \hat{z}, \hat{x}, \hat{y}) = (\bar{s}, \bar{z}, \mathbf{0}, \bar{y})$. This solution contains a set of open hubs, a set of served nodes, and a set of active hub edges. Given that $L_y(\mu)$ and $L_h(\lambda, \mu)$ are independently solved, some hub edges could be associated with closed hub nodes. In order to guarantee the feasibility of the edge variables $\hat{y}$, we close all hub edges that do not have both end-nodes open as hubs. That is, for each $e = \{i, j\} \in E_H$ such that $\hat{z}_i = 0$ or $\hat{z}_j = 0$, we set $\hat{y}_e = 0$. Finally, we select the set of commodities to be served and their routing paths as follows. For each commodity $k \in K$ with both end-nodes activated, we identify the most "attractive" path among the ones using open hub edges (and thus open hub nodes), and route commodity k through it only if it is profitable. That is, for each $k \in K$ with $\hat{s}_{o(k)} + \hat{z}_{o(k)} = \hat{s}_{d(k)} + \hat{z}_{d(k)} = 1$, let $e(k) \in \arg\max\{R_k - F_{ek} : \hat{y}_e = 1, e \in E_H\}$. If $R_k - F_{e(k)k} > 0$, then $\hat{x}_{e(k)k} = 1$, and 0 otherwise.

4.3.2. A Primal Heuristic for PO_2

To obtain feasible solutions to PO_2 we apply a two phase heuristic. The first phase is an adaptation of the heuristic applied to PO_1. Since the quality of the PO_2 solutions produced by such first phase is usually quite weak, we apply a second phase to improve the outcome of the first phase.

The first phase starts with $(\hat{s}, \hat{z}, \hat{x}, \hat{y}) = (\bar{s}, \bar{z}, \mathbf{0}, \bar{y})$, and then closes all hub edges that do not have both end-nodes open as hubs. The set of commodities to be served and their routing paths are selected as follows. In order to satisfy constraints (11), for each commodity with both end-nodes activated we identify the best path among the ones using open hub edges, and route such commodity through it regardless if it is profitable or not. That is, for each $k \in K$ with $\hat{s}_{o(k)} + \hat{z}_{o(k)} = \hat{s}_{d(k)} + \hat{z}_{d(k)} = 1$, let $e(k) \in \arg\max\{R_k - F_{ek} : \hat{y}_e = 1, e \in E_H\}$ and set $\hat{x}_{e_k k} = 1$ (independently of the sign of $R_k - F_{e_k k}$). Let $\hat{\eta}$ denote the objective value of $(\hat{s}, \hat{z}, \hat{x}, \hat{y})$.

The second phase is a three-step procedure that aims at improving the output of Phase 1 by: (i) activating additional hub edges, (ii) adding new served nodes, and (iii) closing open hub nodes.

(i) For each non-activated hub edge $e = \{i, j\} \in E_H$ but with both endnodes open as a hubs, we compute the variation in the objective function if hub edge e were activated and the commodities re-routed accordingly. Then, the hub edge is activated if the estimation is positive. That is, we consider in an arbitrary order each $e = \{i, j\} \in E_H$ with $\hat{y}_e = 0$ and $\hat{z}_i = \hat{z}_j = 1$, and for each $k \in K$ we set

$$\Delta_k = \begin{cases} \max\{R_k - F_{ijk}, 0\} & \text{if } \sum_{(i',j') \in A} \hat{x}_{i'j'k} = 0, \\ \max\{F_{e_k k} - F_{ijk}, 0\} & \text{if } \hat{x}_{e_k k} = 1, \\ 0 & \text{otherwise.} \end{cases}$$

If $\Gamma_e = \sum_{k\in K} \Delta_k - r_{ij} > 0$, then $\hat{y}_e = 1$ and $\hat{\eta} = \hat{\eta} + \Gamma_e$.

(ii) For each node $i \in N$ that is not served, we compute the variation in the objective function if node i was served and its associated commodities routed. The node is then served if the estimation is positive. We denote as $\hat{A} = \{(i', j') \in A \mid \hat{y}_{i'j'} = 1\}$ the set of arcs whose associated hub edges are active in the current solution. We consider in an arbitrary order each $i \in N$ with $\hat{s}_i = 0$, and for each $k \in K$ we define

$$\Delta_k = \begin{cases} \max\{R_k - \min_{(i',j')\in\hat{A}}\{F_{i'j'k}\}, 0\}, & \text{if } o(k) = i \text{ and } \hat{s}_{d(k)} + \hat{z}_{d(k)} = 1, \\ \max\{R_k - \min_{(i',j')\in\hat{A}}\{F_{i'j'k}\}, 0\}, & \text{if } d(k) = i \text{ and } \hat{s}_{o(k)} + \hat{z}_{o(k)} = 1, \\ 0 & \text{otherwise.} \end{cases}$$

If $\Gamma_i = \sum_{k\in K} \Delta_k - c_i > 0$, then $\hat{s}_i = 1$ and $\hat{\eta} = \hat{\eta} + \Gamma_i$.

(iii) For each hub $i \in H$ that is open we compute the variation in the objective function if hub i was closed and its associated commodities re-routed. The hub node is then closed if the estimation is positive. We denote as $\hat{E}(i) = \{\{i', j'\} \in E \mid \hat{y}_{i'j'} = 1, \text{ and } i' = i \text{ or } j' = i\}$ the set of active hub edges incident to i. We consider in an arbitrary order each $i \in N$ with $\hat{z}_i = 1$, and for each $k \in K$ we define

$$\Delta_k = \begin{cases} -(R_k - F_{i(k)j(k)k}), & \text{if } \hat{x}_{i(k)j(k)k} = 1 \text{ and } \{i(k), j(k)\} \in \hat{E}(i), \\ 0, & \text{otherwise.} \end{cases}$$

If $\Gamma_i = \sum_{k\in K} \Delta_k + f_i + \sum_{(i',j')\in\hat{E}(i)} r_{i'j'} y_{i'j'} > 0$, then $\hat{z}_i = 0$, $\hat{y}_e = 0$ for all $e \in \hat{E}(i)$, and $\hat{\eta} = \hat{\eta} + \Gamma_i$.

5. Variable Elimination Techniques

One of the main challenges of the MIP formulations we use to model PO_1 and PO_2 are the very large number of variables and constraints that these require, even for small-size instances. By slightly increasing the size of the instances, the number of variables in the formulations becomes so large that considerable amounts of computing time and memory are required to solve them with a commercial solver. In the previous sections, we have presented LRs whose Lagrangean functions can be solved efficiently in polynomial time. Still, any reduction on the size of the formulations is highly beneficial for attaining a higher efficiency. In our algorithmic framework we reduce the size of the instances by means of three effective procedures: (i) *Preprocessing*, valid only for PO_1, which is applied prior to the solution of D_1, and aims at eliminating variables and constraints; (ii) *Reduction Tests*, valid for both PO_1 and PO_2, which eliminate variables based on the information obtained from the Lagrangean functions; and, (iii) *Post-processing*, which further eliminates variables, both for PO_1 and PO_2, using jointly information from the reduction tests and valid lower bounds.

5.1. Preprocessing

In the case of PO_1, it is possible to a priori eliminate routing variables x that will not make part of an optimal solution by using the following property.

Property 1. [*Alibeyg et al. (2016)*] *There is an optimal solution to formulation* (1) – (10) *where $x_{ijk} = 0$, for all $k \in K$ and $(i, j) \in A_H$, with $R_k - F_{ijk} \leq 0$.*

The use of Property 1 in PO_1 allows to eliminate all routing variables with unprofitable arcs. That is, for each $k \in K$ we set $x_{ijk} = 0$ for all $(i, j) \in A_H$ such that $R_k - F_{ijk} \leq 0$. Since we are assuming that routing costs are symmetric, if $(i, j) \in A_H$ is unprofitable so is $(j, i) \in A_H$. Thus, when we set $x_{ijk} = 0$, we not only set $x_{jik} = 0$, but also eliminate the corresponding constraint (6), as it becomes unnecessary. Hence, for each $k \in K$ we restrict the set of potential candidate arcs for routing it to the arcs that are profitable for this commodity, $A_k = \{(i, j) \in A_H \mid R_k - F_{ijk} > 0\}$. Let also E_k denote the corresponding set of profitable hub edges for k.

Since the above elimination affects variables and constraints of PO_1, it can also be extended to the Lagrangean function $L_1(\lambda, \mu)$, where only arcs and edges of A_k and E_k, respectively, will now be considered. We also note that the reduction on the number of constraints (6) of PO_1 causes a significant reduction on the number of Lagrangean multipliers μ in $L_1(\lambda, \mu)$.

An important consequence of (11), is that Property 1 does not hold for PO_2 as all the commodities whose O/D nodes are active must be served, independently of whether or not there are profitable arcs for them.

5.2. Reduction Tests

Another way of reducing the size of the formulations is to develop tests to eliminate variables based on information generated from the LR. We next develop two such tests based on sufficient conditions that determine if a potential hub will be closed or if a hub edge will not be activated in an optimal solution of a given HNDPP instance. These tests are valid for both PO_1 and PO_2, since they are based on the information produced by their respective Lagrangean functions L_1 and L_2. We will not distinguish the case of PO_1 from the case of PO_2, since the structure of the terms that constructs the Lagrangean functions $L_1(\lambda, \mu)$, and $L_2(\lambda, \mu)$ is exactly the same and the rationale of the tests is also the same in both cases. Similar reduction tests have been successfully applied to other HLPs (see Contreras et al., 011a,b).

5.2.1. Elimination of Potential Hub Nodes

The idea of this test is to use the Lagrangean function to obtain upper bounds on the profit that would be obtained in the original problem if a given node $l \in H$ is chosen to become a hub. If this estimated profit is less than the value of the best known solution to the original problem, then node l will not be a hub in any optimal solution. Let $\hat{L}_h(\lambda, \mu, S_z)$ denote the value of $L_h(\lambda, \mu)$ when restricted to a set of potential hub nodes $S_z \subseteq H$, and its associated set of hub arcs $A_S = \{(i,j) \in A_H : i, j \in S_z\}$. That is,

$$\begin{aligned} \hat{L}_h(\lambda, \mu, S_z) = \text{maximize} \quad & \sum_{k \in K} \overline{Q}_k h_{o(k)} h_{d(k)} - \sum_{i \in N} F_i h_i \\ \text{subject to} \quad & h_i \in \{0,1\} \qquad i \in N, \end{aligned}$$

where $\overline{Q}_k = \max\left\{0, \max_{(i,j)\in A_S}\left\{\overline{P}_{ijk}\right\}\right\}$. Let $\hat{L}_h^l(\lambda, \mu, S_z)$ denote the optimal value of $\hat{L}_h(\lambda, \mu, S_z)$ with the additional constraint that hub l is open, i.e. $z_l = 1$. The only difference between $\hat{L}_h(\lambda, \mu, S_z)$ and $\hat{L}_h^l(\lambda, \mu, S_z)$ is that, in the latter, node l is now a priori activated as an open hub. This means that now $F_l = \{f_l\}$ and $h_l = 1$. The following result can be used to perform variable elimination tests on hub location decisions.

Proposition 2. *Let $\underline{\eta}$ be a valid lower bound on the optimal value of PO_1 (resp. PO_2), $S_z \subseteq H$ a given set of potential hub nodes, $l \in S_z$ a specific potential hub node, and (λ, μ) a multiplier vector. If $\Delta_l(\lambda, \mu, S_z) = L_y(\mu) + \hat{L}_h^l(\lambda, \mu, S_z) < \underline{\eta}$, then $z_l = 0$ in any optimal solution.*

Proof The result follows since $\Delta_l(\lambda, \mu, S_z)$ is an upper bound on the objective function value of any solution in which a hub is located at node l. Therefore, if $\Delta_l(\lambda, \mu, S_z) < \underline{\eta}$, no optimal solution will have an open hub at $l \in S_z$, so $z_l = 0$. ■

We use this result as follows. The subgradient optimization is initialized with all possible nodes as candidate hub nodes, that is $S_z = H$. Once the deviation between the upper and lower bounds becomes smaller than a given threshold ϵ_{Test} after a number of iterations of the subgradient optimization algorithm, we apply the reduction test for each $l \in S_z$ that is not active in the current subgradient optimization iteration, i.e. $\bar{s}_l = \bar{z}_l = 0$, every $niter_{\text{Test1}}$ iterations. If $\Delta_l(\lambda, \mu, S_z) < \underline{\eta}$, we eliminate l from the set of candidate hub nodes, i.e. $S_z \leftarrow S_z \setminus \{l\}$. According to Proposition 2, by applying the test in this way we ensure that S_z always contains an optimal set of hubs.

When some node is eliminated from S_z, not only the associated z_l variable is eliminated from the LR, but also several routing variables x_{ijk} associated with node l. This plays an important role in the computational complexity for solving $L_{z,s,x}(\lambda, \mu)$, as the running time is now dependent of the size of A_S, instead of A_H. That is, the Lagrangean functions $L_1(\lambda, \mu)$ and $L_2(\lambda, \mu, \pi)$ can now be solved in $O(|K||A_S| + |N|^3)$ time. Another important consequence of eliminating one variable z_l is that we can remove $|K|$ constraints (5) from the solution process, which in turn significantly reduces the solution space of the Lagrangean dual problems D_1 and D_2.

5.2.2. Elimination of Potential Hub Edges

An immediate consequence of the elimination of potential hub nodes is that if the two end-nodes of a hub edge have been eliminated, then the hub edge can also be eliminated. That is, we set $y_e = 0$ for all $e = \{i, j\} \in E_H$ where z_i and z_j have been set to zero.

Additional hub edges can be further eliminated by estimating an upper bound on the objective function value if a hub edge is activated. This bound can be easily computed after setting at value one the variable associated with the candidate edge in $L_y(\mu)$. In particular, for a set of candidate hub edges $S_y \subseteq E_H$, and a hub edge $\bar{e} \in S_y$, let $\hat{L}_y^{\bar{e}}(\mu, S_y)$ denote the optimal value of $L_y(\mu)$ restricted to S_y when hub edge $\bar{e}$ has been activated

$$\hat{L}_y^{\bar{e}}(\mu, S_y) = -\overline{r}_{\bar{e}} - \sum_{e \in S_y \setminus \{\bar{e}\}} \min\{0, \overline{r}_e\}.$$

The following result can be used to perform reduction tests on hub edge activation decisions.

Proposition 3. *Let $\underline{\eta}$ be a valid lower bound on the optimal value of PO_1 (resp. PO_2), $S_y \subseteq E_H$ a given set of potential hub edges, $\bar{e} \in S_y$ a spe-*

cific potential hub edge, and (λ, μ) a multipliers vector. If $\Delta_{\bar{e}}(\lambda, \mu, S_y) = \hat{L}_y^{\bar{e}}(\mu, S_y) + L_{z,s,x}(\lambda, \mu) < \underline{\eta}$, then $y_{\bar{e}} = 0$ in any optimal solution.

Proof The result follows since $\Delta_{\bar{e}}(\lambda, \mu, S_y)$ is an upper bound on the objective function value of any solution in which a hub edge $\bar{e}$ is activated. Therefore, if $\Delta_{\bar{e}}(\lambda, \mu, S_y) < \underline{\eta}$, hub edge $\bar{e}$ will not be activated in any optimal solution. ■

Reduction tests for hub edges are applied immediately after reduction tests for hub nodes. Let E_{H^0} denote the set of edges eliminated in the first phase of the hub elimination test. For the second phase, we set $S_y = E_H \backslash E_{H^0}$, and apply the elimination test to each candidate hub edge $\bar{e}$ in the updated set S_y. Then, if $\Delta_{\bar{e}}(\lambda, \mu, S_y) < \underline{\eta}$, we eliminate $\bar{e}$ from the set of candidate hub edges, i.e. $S_y \leftarrow S_y \setminus \{\bar{e}\}$. According to Proposition 3, applying the test in this way ensures that S_y always contains an optimal set of hub edges.

In addition, once a y_e variable has been eliminated, we can also remove $|K|$ constraints (6) from the solution process, which causes a considerable reduction of the solution space of the Lagrangean dual problems D_1 and D_2.

5.3. Post-processing

This is a simple procedure where we use information obtained from the reduction tests for hub edges to update the set of candidate hub edges A_k, so as to further eliminate additional routing variables x_{ijk}. In particular, for each $k \in K$, we remove from its set of profitable edges A_k any hub edge that has been fixed to zero during the hub edge elimination test. That is, any variable x_{ijk} associated with an arc removed from A_k is permanently set at value 0. Given that the amount of time for updating this set is significant, this procedure is only applied every $niter_{\text{Test2}}$ applications of the tests.

Algorithm (2) summarizes the enhanced subgradient optimization algorithm with reduction tests. Note that the primal heuristic described in the previous section is applied every $niter_{heur}$ iterations of the subgradient algorithm.

6. An Exact Solution Algorithm

In this section we present the complete algorithmic framework used for solving problems PO_1 and PO_2 to optimality. Its core component is a branch-and-bound method in which, at every node of the enumeration tree, we obtain

Algorithm 2 Subgradient Optimization with Reduction Tests

```
Initialization
    Initialize $Z_{D_1} \leftarrow \infty$; $S_z \leftarrow H$; $S_y \leftarrow E_H$; $(\lambda^0, \mu^0)$; $\Lambda^0$
    Let $\underline{\eta}$ be a lower bound on the optimal solution value
    Apply preprocessing to initialize $A_k$ and $E_k$ sets
while Stopping criteria not satisfied do
    Solve $L_1(\lambda^t, \mu^t, S_z, S_y)$ and obtain an optimal solution $(\bar{s}, \bar{z}, \bar{x}, \bar{y})$
    if $L_1(\lambda^t, \mu^t, S_z, S_y) < Z_{D_1}$ then
            $Z_{D_1} \leftarrow L_1(\lambda^t, \mu^t, S_z, S_y)$
    end if
    if $t \mod niter_{heur} = 0$ then
        Apply primal heuristic and update $\underline{\eta}$
    end if
    if $(|Z_{D_1} - \underline{\eta}|/\underline{\eta}) \times 100 < \epsilon_{test}$ then
        if $t \mod niter_{\text{Test1}} = 0$ then
            for $l \in S_z$ do
                if $\Delta_l(\lambda, \mu, S_z) < \underline{\eta}$ then
                    $S_z \leftarrow S_z \setminus l$
                end if
            end for
            for $\bar{e} \in S_y$ do
                if $\Delta_{\bar{e}}(\lambda, \mu, S_y) < \underline{\eta}$ then
                    $S_y \leftarrow S_y \setminus \bar{e}$
                end if
            end for
        end if
        if $t \mod niter_{\text{Test2}} = 0$ then
            Apply post-processing to update $A_k$ sets
        end if
    end if
    Compute the subgradient $\gamma^t$
    Compute the step length $\varepsilon_t \leftarrow \Lambda^t(L_1(\lambda^t, \mu^t) - \underline{\eta})/||\gamma^t||^2$
    $(\lambda^{t+1}, \mu^{t+1}) \leftarrow (\lambda^t, \mu^t) + \varepsilon_t \gamma^t$
    $t \leftarrow t + 1$
end while
```

lower and upper bounds by using the subgradient optimization algorithms and the primal heuristics presented in Section 4. We also apply a partial enumeration phase to enhance the application of the reduction tests. This phase is applied at the beginning of the branch-and-bound procedure right after solving the root node. It is particularly useful to reduce the number of variables to branch on, and to reduce the size of the subproblems in the nodes of the tree. Contreras et al. (011a,b) provide some examples of successful implementations of branch-and-bound algorithms based on Lagrangean bounds used to solve HLPs. We next describe the partial enumeration and then the overall branch-and-bound algorithm.

6.1. Partial Enumeration

The partial enumeration works as follows. Let H^0 and H^1 denote the set of potential hubs that have been already fixed at value 0 and 1, respectively. Since, the partial enumeration is applied after solving the root node, initially we have $H^0 = H \setminus S_z$ and $H^1 = \emptyset$. Then, for each hub not yet considered $i \in H \setminus (H^0 \cup H^1)$, we temporarily fix $z_i = 1$ and solve the resulting Lagrangean dual problem using an iteration limit of $Iter_{max} = 80$. If the resulting upper bound ub_i^1 is smaller than the current best lower bound, we set $z_i = 0$ (as well as the the related y variables) and we update the set H^0, accordingly. Otherwise, we temporarily fix $z_i = 0$ and solve the resulting Lagrangean function. If the obtained upper bound ub_i^0 is smaller than the current best lower bound, we set $z_i = 1$ and update the set H^1. At the end of the partial enumeration we re-optimize the Lagrangean dual problem using an iteration limit of $Iter_{max} = 1,000$ to further improve the bound of the root node. The partial enumeration phase is summarized in Algorithm 3.

6.2. Branch and Bound

We now present a branch-and-Bound algorithm in which valid lower and upper bounds are constructed at each node of the enumeration tree with the proposed LR. The tree is structured in three levels: the first level where we branch on the z variables (hub nodes); the second level where we branch on the s variables (served nodes); and a third level, where we branch on the y variables (hub edges). Each level is explored according to a depth first search policy in which the 1-branch is explored first. No subsequent level is explored until all the nodes of the previous level have been explored. We use three standard rules for pruning nodes: i) pruning by *infeasiblity* (empty partition), ii) pruning by *optimality* (the optimal solution to the Lagrangean

Algorithm 3 Exact Algorithm for PO_1: Partial Enumeration Phase

```
Require: 1: (Solve root node)
  Solve $D_1$ with Algorithm 2
  Initialize $\overline{\eta} \leftarrow Z_{D_1}$; $H^0 \leftarrow H \setminus S_z$; $H^1 = \emptyset$
  Let $\underline{\eta}$ be the best lower bound obtained with Algorithm 2
Require: 2: (Partial enumeration)
  for $i \in H \setminus (H^0 \cup H^1)$ do
      Solve $D_1$ with $z_i = 1$ to obtain $ub_i^1$
      if $ub_i^1 < \underline{\eta}$ then
          $H^0 \leftarrow H^0 \cup i$
      else
          Solve $D_1$ with $z_i = 0$ to obtain $ub_i^0$
          if $ub_i^0 < \underline{\eta}$ then
              $H^1 \leftarrow H^1 \cup i$
          end if
      end if
  end for
  Apply post-processing to update $A_k$ sets
  Resolve $D_1$ with Algorithm 1 to update $\overline{\eta}$
  if $(|\overline{\eta} - \underline{\eta}|/\underline{\eta}) \times 100 > \epsilon$ then
      Go to Step 3
  else
      Stop with an $\epsilon$-optimal solution
  end if
```

dual is feasible for the original problem), and iii) pruning by *bound* (upper bound of node is strictly smaller than best known lower bound).

The strategy for selecting of the branching variable at each node of the first level is guided by the output of the partial enumeration. In particular, for each potential hub node not yet fixed $i \in H \setminus (H^0 \cup H^1)$, we compute $\delta_i = \min\{ub_i^0, ub_i^1\}$. At any point during the first level, the branching variable z_j is the selected as $j \in \arg\max\{\delta_i \mid i \in H \setminus (H^0 \cup H^1)\}$.

After finishing branching on the z variables, we continue branching on the s variables. For each *active* node at the end of the first level, we set $s_i = 0$ for all $i \in H^1$, and continue branching on the remaining s_i variables with $i \in N \setminus H^1$. During the second level, branching variables are arbitrarily selected. If some nodes remain active after completing the branching on the

z and s variables, then the branching on the hub edge variables y begins. For each active node at the end of the second level, we set $y_e = 0$ for all $e \in H^0 \times H^0$. During the third level, branching variables are also arbitrarily selected. Given that the Lagrangean dual problems are only approximately solved with Algorithm 1, it may happen that there are some active nodes after finishing branching in the third level. In this case, the remaining routing subproblems can be efficiently solved to optimality as described in Section 4.3. Finally, at each node of the enumeration tree, we use the optimal dual solution to the Lagrangean dual of its parent node, as the initial solution to the current Lagrangean dual, instead of starting from scratch. The overall branch-and-bound phase is summarized in Algorithm 4.

7. Computational Experiments

We have run extensive computational experiments to analyze and compare the performance of the Lagrangean relaxation, the reduction tests and the exact algorithm, both for PO_1 and PO_2. All algorithms were coded in C and run on an HP station with an Intel Xeon CPU E3-1240V2 processor at 3.40 GHz and 24 GB of RAM under Windows 7 environment. In all the experiments the maximum CPU time was set to 86,400 seconds (one day).

The benchmark instances we have used for our computational study are the same we used in Alibeyg et al. (2016). Most of the data comes from the well-known CAB data set of the US Civil Aeronautics Board and has been obtained from *http://www.researchgate.net/publication/269396247_cab100_mok*. This data provides Euclidean distances d_{ij} between 100 cities in the US and the values of the service demand W_k between each pair of cities. We have considered instances with $n \in \{25, 30, 40, 50, 60, 70, 80, 90, 100\}$ and $\alpha \in \{0.2, 0.5, 0.8\}$. Since the CAB instances do not provide setup costs f_i for opening hubs, we use the ones generated by de Camargo et al. (2008). For the remaining missing information, we use the following additional data that we generated for the computational experiments of Alibeyg et al. (2016). The setup costs c_i for served nodes are $c_i = \nu f_i$, where $\nu = 0.1$ unless otherwise stated. The setup costs for activating hub edges are $r_e = \tau(f_i + f_j)/2$, where $\tau \in \{0.3, 0.6, 0.4\}$ is a parameter used to model the increase (decrease) in setup costs on the hub edges when considering smaller (larger) discount factors α. The revenues R_k for routing commodities are randomly generated as $R_k = \varphi \sum_{(i,j) \in A_H} F_{ijk}/|A_H|$, where φ is a continuous random variable following a uniform distribution $\varphi \sim U\,[0.25, 0.35]$. The collection and distribution factors are $\chi = \delta = 1$.

7.1. Implementation Details

After some fine-tuning, we set the following parameter values for the subgradient optimization algorithm. The maximum number of iterations, $Iter_{max}$, is 3,000 at the root node, 80 at each application of the partial enumeration, and 1,000 in the re-optimization after the partial enumeration. At each node of the branch and bound tree we set $Iter_{max} = 200$. The additional parameters that are used for the termination criteria of the subgradient optimization are the following: the threshold between the upper and lower bounds is $\epsilon = 10^{-6}$ (termination criterion *ii*); and the number of consecutive iterations without improvement is $niter = 1,500$ (termination criterion *iii*). We set $(\lambda^0, \mu^0, \pi^0) = (95, 85, 85)$ as the initial multipliers vector. The parameter Λ^t that is used in the computation of the step length is initialized to 7 and halved every 500 iterations, provided that the % gap is less than 50%, and is reset to its initial value whenever it becomes smaller than 2. We apply the heuristics every 10 iterations of the subgradient algorithm. We use $\underline{\eta} = 0$ as the initial lower bound. This value is updated and recorded for further applications of the subgradient and the elimination tests, whenever the heuristic improves the incumbent solution. We apply the elimination tests every $niter_{\text{Test1}} = 100$ and iterations of subgradient optimization and the post-processing every $niter_{\text{Test2}} = 700$ applications of the tests. Both the tests and post-processing are only applied if the percentage gap between the upper and lower bounds is below the threshold $\epsilon_{\text{Test}} = 5\%$.

7.2. Comparison of the Exact Algorithmic Framework and CPLEX

We next analyze and compare the performance of the general purpose solver CPLEX 12.6.3 using a traditional (deterministic) branch-and-bound algorithm and our exact algorithmic framework for PO_1 and PO_2. The application of CPLEX to PO_1 and PO_2 is referred to as $CPLEX_1$ and $CPLEX_2$, respectively, whereas our exact algorithms for PO_1 and PO_2 are referred to as BB_1 and BB_2, respectively. All parameters have been set to their default values both in $CPLEX_1$ and $CPLEX_2$. It is worth mentioning that, similar to Alibeyg et al. (2016), Property 1 is also applied to $CPLEX_1$.

Figures 1 and 2 give performance profiles of $CPLEX_1$ (dotted line) and BB_1 (solid line), and of $CPLEX_2$ (dotted line) and BB_2 (solid line), respectively. In each figure, the horizontal axis refers to computing times while the vertical axis refers to number of instances. The points (x, y) depicted in the lines on each figure indicate the total number of instances y optimally solved within the computing time x. In general, small size instances can be solved rather fast both with CPLEX and our exact algorithms, but the performance decreases as the sizes of the instances increase. This is why in the two lines depicted in each figure the vertical values increase fast at the beginning but slow down after a while. Throughout the considered one-day time interval, BB_1 is consistently better than $CPLEX_1$. Moreover, within the time limit, BB_1 is able to optimally solve all 27 instances, whereas $CPLEX_1$ only solves 18. The effect of the additional set of constraints (11) on the difficulty for solving PO_2 is evident, and both $CPLEX_2$ and BB_2 are slower than their respective counterparts for PO_1. In any case, BB_2 still outperforms $CPLEX_2$ and, within the time limit, it is able to optimally solve 21 instances instead of the 15 instances optimally solved by $CPLEX_2$.

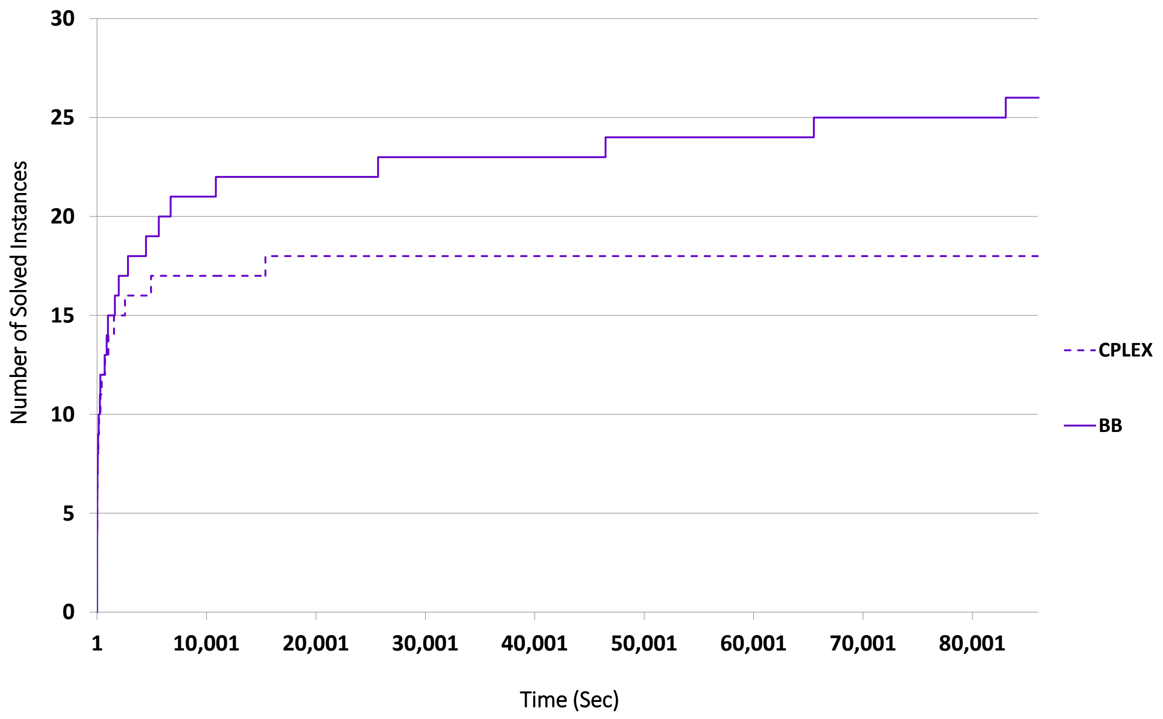

Figure 1: Performance profile of CPLEX and BB_1 for PO_1.

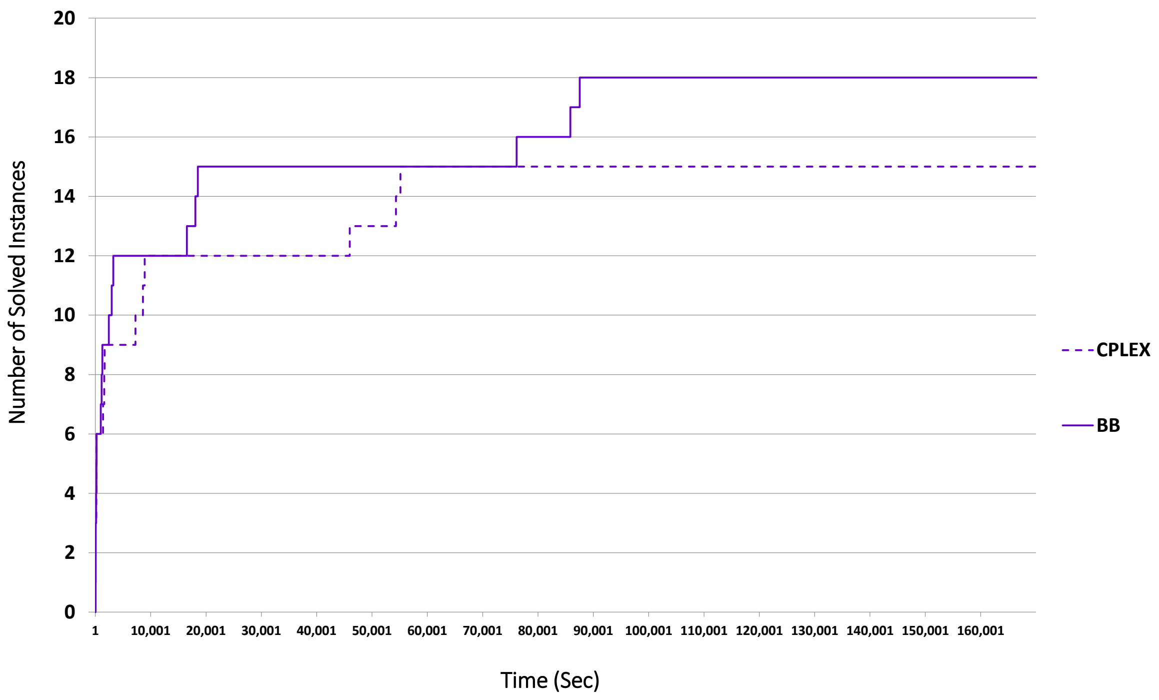

Figure 2: Performance profile of CPLEX and BB_2 for PO_2.

Tables 1 and 2 give information of the bounds at the root nodes and of the complete enumeration trees of the compared solution methods for PO_1 and PO_2, respectively. The first two columns of each table give some instances data: α, the discount factor on hub edges, and $|N|$, the number of nodes. The next two columns, under the heading *% Dev*, give the percentage deviations of the upper bounds produced by the employed relaxations: Linear Programming (LP) in the case of CPLEX and Lagrangean in our proposed solution algorithms. These deviations have been computed as $100(v_{RP} - v^*)/v^*$, where v_{RP} denotes the upper bound produced by the relaxed problem (LP or Lagrangean) and v^* the optimal or best-known value. The next two columns under the header *Nodes* give the number of nodes explored in the enumeration trees. The three columns under the header *Time (sec)* give computing times in seconds. The first of these columns gives the total time consumed by CPLEX, and the other two refer to our exact solution algorithms: *LR* for the computing time for solving the Lagrangean Dual at the root node and *BB* for the overall time needed to optimally solve each instance. Finally, the last two columns *RT* and *PE* give the percentage of hubs fixed with the reduction tests (see Section 5.2) and with the partial enumeration (see Section 6.1), respectively. That is, the entries of these columns are computed as $100(FH/|H|)$, where FH is the number of hubs fixed in each case. The entries corresponding to instances that could not be handled by CPLEX because of insufficient memory are filled with the text *mem*. When an instance could not be solved to optimality within the time limit, the corresponding entry in the column of the computing times is *time* followed by the percentage optimality gap at termination, in parenthesis.

The results of Table 1 confirm the superiority of BB_1 over $CPLEX_1$. On the one hand, even if formulation (1)-(10) produces, in general, very tight LP bounds, it has a very strict limitation in terms of the size of the instances that can be handled by $CPLEX_1$. It is true that the LP gap of $CPLEX_1$ is always 0.00% for the 18 instances with up to 70 nodes. However, the quality of these bounds contrasts with the insufficiency of the 24 GB of memory available: none of the remaining nine instances with 80-100 nodes could even be uploaded to the CPLEX solver. In contrast, our Lagrangean Dual D_1 is highly effective in all cases, as it is able to produce tight bounds for all 27 instances using only 2 GB of memory for the largest considered instances with up to 100 nodes. In some cases achieving convergence when solving D_1 was very difficult, and the actual upper bound Z_{D_1} could not be attained. This explains why in some cases % *Dev* is 0.00 for *LP*, but it is strictly positive for *LR*. Still, the bounds we could obtain with D_1, together with the quality of the heuristic applied within subgradient optimization, assess its effectiveness. The optimality of four out of the 27 PO_1 instances was already proven after solving D_1 at the root node. For these instances, the heuristic applied within subgradient optimization produced a feasible solution with the same value as that of the upper bound. For 16 and seven of the remaining 23 instances, the percent deviation after solving D_1 was below 0.5% and 1.46%, respectively.

The columns under *Time (sec)* relative to D_1 and BB_1 confirm that these good results were obtained with a small computing effort. On the one hand, BB_1 is able to solve all 27 instances to proven optimality within the CPU time limit, while $CPLEX_1$ is able to solve only instances with up to 70 nodes. On the other hand, BB_1 is, in general, much faster than $CPLEX_1$ on the 18 instances that could be solved by $CPLEX_1$, particularly for the instances

α	$\|N\|$	*% Dev*		*Nodes*		*Time (sec)*			*% Fixed hubs*	
		LP	*LR*	*CPLEX*	*BB*	*CPLEX*	*LR*	*BB*	*RT*	*PE*
0.2	25	0.00	0.00	0	0	3.00	1.80	1.80	0.00	0.00
	30	0.00	0.00	0	0	9.64	5.63	5.63	0.00	0.00
	40	0.00	0.04	0	0	126.12	35.78	45.66	0.00	100.00
	50	0.00	0.11	0	0	513.20	81.94	120.31	0.00	100.00
	60	0.00	0.16	0	160	2370.97	181.69	349.34	0.00	98.33
	70	0.00	0.27	0	218	10460.44	391.55	850.60	0.00	98.57
	80	mem	0.26	mem	340	mem	736.90	1641.43	0.00	98.75
	90	mem	0.36	mem	1318	mem	1298.37	4129.42	1.11	97.78
	100	mem	0.64	mem	6738	mem	1970.64	19048.79	0.00	90.00
0.5	25	0.00	0.00	0	0	1.60	0.36	0.36	8.00	8.00
	30	0.00	0.00	0	0	4.55	3.85	3.85	13.33	13.33
	40	0.00	0.03	0	0	21.53	13.67	19.08	15.00	100.00
	50	0.00	0.10	0	0	75.90	45.63	59.83	18.00	100.00
	60	0.00	0.13	0	162	309.06	110.09	189.35	18.33	98.33
	70	0.00	0.27	0	570	1006.20	248.81	626.97	7.14	97.14
	80	mem	0.40	mem	600	mem	446.79	1170.51	12.50	92.50
	90	mem	1.46	mem	3676	mem	634.52	14824.07	0.00	67.78
	100	mem	1.28	mem	3666	mem	1161.68	17537.49	1.00	77.00
0.8	25	0.00	0.02	0	0	1.36	1.83	2.34	24.00	100.00
	30	0.00	0.01	0	0	3.62	3.42	4.71	20.00	100.00
	40	0.00	0.03	0	0	15.42	9.61	13.55	22.50	100.00
	50	0.00	0.36	0	128	44.93	22.09	43.13	24.00	94.00
	60	0.00	0.27	0	132	121.70	46.91	100.51	25.00	96.67
	70	0.00	0.51	0	166	293.20	155.21	386.88	2.86	85.71
	80	mem	0.53	mem	792	mem	267.48	1085.56	5.00	88.75
	90	mem	0.88	mem	24214	mem	471.32	20789.11	6.67	88.89
	100	mem	0.87	mem	52372	mem	698.48	58546.99	11.00	89.00

Table 1: Results of exact algorithm using CAB instances for PO_1

with the smallest discount factor $\alpha = 0.2$. Note that BB_1 is faster than $CPLEX_1$ in 15 of out of the 18 such instances. Finally, the last two columns of Table 1 assess the effectiveness of the reduction tests and, particularly, of the partial enumeration: in 21 benchmark instances it was possible to fix more than 80% of the hubs. The side effect of the good performance of these tests is that no enumeration is required in 11 out of the 27 tested instances.

The results of Table 2 confirm that, as mentioned, solving PO_2 is more

α	$\lvert N\rvert$	*% Dev*		*Nodes*		*Time (sec)*			*% Fixed hubs*	
		LP	*LR*	*CPLEX*	*BB*	*CPLEX*	*LR*	*BB*	*RT*	*PE*
0.2	25	0.00	0.00	0	0	25.15	28.23	28.23	12.00	12.00
	30	0.00	0.07	0	30	130.19	61.34	81.31	13.33	93.33
	40	0.00	0.20	0	166	1162.89	486.23	735.45	2.50	95.00
	50	0.00	0.19	0	150	5557.70	552.99	1153.80	4.00	98.00
	60	0.00	0.71	0	872	37065.80	7015.70	13311.15	0.00	83.33
	70	mem	0.97	mem	2610	mem	7550.15	57608.85	0.00	64.29
	80	mem	1.15	mem	259	mem	15347.87	time (0.02)	0.00	0.00
	90	mem	1.40	mem	335	mem	27938.78	time (0.92)	43.33	47.77
	100	mem	1.44	mem	573	mem	14092.6	time (0.35)	0.00	64.00
0.5	25	0.00	0.04	0	0	10.24	21.04	23.32	40.00	100.00
	30	0.00	0.03	0	0	31.81	41.99	46.45	36.67	100.00
	40	0.00	0.14	0	0	216.29	138.28	162.76	37.50	100.00
	50	0.00	0.29	0	100	1364.90	379.20	530.78	42.00	94.00
	60	0.00	0.39	0	0	8339.37	1326.02	1526.58	36.67	100.00
	70	mem	0.87	mem	524	mem	3654.87	9727.55	27.14	75.71
	80	mem	0.91	mem	622	mem	7117.06	15128.05	0.00	85.00
	90	mem	4.02	mem	175	mem	10881.89	time (2.77)	0.00	12.22
	100	mem	3.74	mem	54	mem	18334.13	time (3.17)	0.00	17.00
0.8	25	0.00	0.04	0	0	7.06	18.40	19.83	44.00	100.00
	30	0.00	0.02	0	0	17.87	25.64	28.32	50.00	100.00
	40	0.00	0.05	0	34	88.42	125.79	141.74	45.00	95.00
	50	0.00	0.11	0	0	305.75	252.98	280.06	52.00	98.00
	60	0.00	0.09	0	0	805.18	420.89	453.19	53.33	100.00
	70	mem	0.38	mem	198	mem	1259.86	1690.30	50.00	97.14
	80	mem	0.68	mem	220	mem	3931.77	5099.68	42.50	93.75
	90	mem	1.05	mem	8366	mem	5122.81	83735.91	0.00	93.33
	100	mem	1.04	mem	11174	mem	8995.96	time (0.61)	42.00	88.00

Table 2: Results of the exact algorithm for PO_2 with CAB instances

challenging than solving PO_1 both for CPLEX and for our exact algorithmic framework. In any case, the superiority of our exact algorithm over CPLEX becomes even more evident for PO_2 than for PO_1. In particular, with the 24 GB of memory available, $CPLEX_2$ could only handle the 15 instances with up to 60 nodes, all of which were optimally solved at the root node. However, it was not possible to even upload to CPLEX any of the remaining 12 instances with 70-100 nodes. The reason for which $CPLEX_2$ could handle fewer instances than $CPLEX1$ is that Property 1 no longer applies to PO_2 so, for a given instance, the actual size formulation (1)-(11) is considerably larger than that of the PO_1 formulation (1)-(10). Despite the fact that Property 1 no longer applies to the PO_2 formulation (1)-(11), D_2 could be optimally solved for all 27 instances using only 3 GB of memory, producing percentage deviations $\%Dev$ smaller than 1% for 20 of the instances, and smaller than 4.02% for the remaining 6 instances. Moreover, BB_2 was able to solve to optimality 21 benchmark instances within the time limit of 86,400 seconds. For the remaining six instances the percentage optimality gaps at termination (given in parentheses under the column *Time (sec)*) never exceed 3.17%. The effectiveness of the partial enumeration and the reduction tests is higher in PO_2 than in PO_1. This effectiveness is particularly noticeable for the instances with higher values of α. Altogether, the partial enumeration was able to fix all the hubs in 7 instances, and the reduction tests fixed more than 40% of the hubs in 11 additional instances.

We complete the information reported and discussed above, by analyzing in detail the performance of each of the steps of the enumeration trees of BB_1 and BB_2. In particular, Tables 3 and 4 show additional information of the partial enumeration at the root node, as well as of each of the branching levels, namely branching on hubs (z variables), branching on served nodes (s variables) and branching on hub edges (y variables). The first two columns in each table give the discount factor α, and the number of nodes $|N|$ of each instance. The next three columns under the heading of *Nodes* depict the exact number of nodes explored at each of the levels of the enumeration trees: enumeration on the hub variables (z), enumeration on the served nodes variables (s), and enumeration on the hub edges variables (y). The next five columns, under the heading *Time (sec)*, indicate the computing times, in seconds, consumed at each of the following steps: root node, partial enumeration, branching on z, branching on s, and branching on y. Similarly, the last four columns under the heading *%Dev* give the percent deviation of the best-known solution at the end of each step relative to the optimal (or best-known solution). These deviations have been computed as $100(v - v^*)/v^*$ where v is the upper bound at the end of each level, and v^* denotes the optimal or best-known value for each instance.

Table 3 further confirms the effectiveness for PO_1 of Property 1 and of the partial enumeration at the root node, which allow fixing hubs and also eliminating hub edges. Note that, particularly for smaller values of α, the enumeration trees of BB_1 generate very few nodes at the first level (z) and

α	$\lvert N\rvert$	*Nodes*			*Time (sec)*					*% Dev*			
		z	s	y	*Root*	PE	z	s	y	*Root*	PE	z	s
0.2	25	0	0	0	2	0	0	0	0	0.00	0.00	0.00	0.00
	30	0	0	0	6	0	0	0	0	0.00	0.00	0.00	0.00
	40	0	0	0	36	9	0	0	1	0.04	0.00	0.00	0.00
	50	0	0	0	82	36	2	0	0	0.11	0.03	0.00	0.00
	60	2	104	54	182	106	7	36	18	0.16	0.06	0.06	0.01
	70	2	146	70	392	322	13	86	38	0.27	0.05	0.05	0.01
	80	2	220	118	737	588	21	202	92	0.26	0.11	0.11	0.02
	90	4	758	556	1298	1049	53	1097	632	0.36	0.19	0.19	0.04
	100	44	3182	3512	1971	2184	418	7607	6870	0.64	0.21	0.20	0.04
0.5	25	0	0	0	0	0	0	0	0	0.00	0.00	0.00	0.00
	30	0	0	0	4	0	0	0	0	0.00	0.00	0.00	0.00
	40	0	0	0	14	4	1	0	0	0.03	0.01	0.01	0.00
	50	0	0	0	46	12	2	0	0	0.10	0.07	0.07	0.00
	60	2	130	30	110	32	5	33	9	0.13	0.04	0.04	0.01
	70	6	452	112	249	129	16	187	46	0.27	0.13	0.13	0.06
	80	28	404	168	447	272	74	288	89	0.40	0.23	0.20	0.02
	90	454	2144	1078	635	1189	3377	6646	2977	1.46	1.02	0.36	0.03
	100	572	2480	614	1162	1428	5028	8033	1886	1.28	0.81	0.22	0.06
0.8	25	0	0	0	2	0	0	0	0	0.02	0.00	0.00	0.00
	30	0	0	0	3	1	0	0	0	0.01	0.00	0.00	0.00
	40	0	0	0	10	3	0	0	1	0.03	0.00	0.00	0.00
	50	8	120	0	22	10	4	8	0	0.36	0.10	0.10	0.00
	60	4	128	0	47	20	6	28	0	0.27	0.11	0.11	0.00
	70	52	114	0	155	108	74	50	0	0.51	0.09	0.02	0.02
	80	70	722	0	267	219	134	465	0	0.53	0.25	0.17	0.00
	90	140	20278	3796	471	473	398	16065	3382	0.88	0.63	0.45	0.45
	100	210	46960	5202	698	677	862	52043	4267	0.87	0.66	0.56	0.56

Table 3: Detailed results of exact algorithm using CAB instances for PO_1

also at the level of the hub edges, where only for 12 out of the 27 instances any such node was generated. As can be seen, the most consuming level is the branching on served nodes (s), but a reduction in the percent deviation can be clearly observed after each step. In any case, the majority of the instances can be solved to optimality in less than one hour of computing time (21 out of 27), including the three larger instances with $N = 80$ nodes, which highlights the efficiency of BB_1.

The results of Table 4 allow making similar observations about the effectiveness of BB_2 for solving PO_2. Similarly to BB_1, there are fewer nodes at the hub nodes level (z) than at the other levels. However, for the largest instances there are still quite a few hubs to branch on after the partial enumeration. Despite the difficulty of PO_2, BB_2 is still robust for solving it:

α	$\|N\|$	*Nodes*			*Time (sec)*					*% Dev*			
		z	s	y	*Root*	PE	z	s	y	*Root*	PE	z	s
0.2	25	0	0	0	28	0	0	0	0	0.00	0.00	0.00	0.00
	30	4	26	0	61	13	3	4	0	0.07	0.06	0.02	0.02
	40	6	106	54	486	86	18	91	55	0.20	0.10	0.10	0.03
	50	2	88	60	553	234	77	171	119	0.19	0.07	0.07	0.03
	60	92	466	314	7016	1536	823	2321	1615	0.71	0.39	0.31	0.05
	70	366	1392	852	7550	4313	10165	21481	14100	0.97	0.44	0.30	0.07
	80	256	3	n.a.	15348	11870	59241	561	time	1.15	1.15	0.02	0.02
	90	335	n.a.	n.a.	27939	17165	41395	time	time	1.40	0.93	0.93	0.92
	100	490	83	n.a.	14093	26171	41851	4778	time	1.44	1.02	0.35	0.35
0.5	25	0	0	0	21	1	1	0	0	0.04	0.02	0.00	0.00
	30	0	0	0	42	3	0	0	0	0.03	0.00	0.00	0.00
	40	0	0	0	138	22	0	0	0	0.14	0.00	0.00	0.00
	50	10	90	0	379	66	23	62	0	0.29	0.12	0.12	0.12
	60	0	0	0	1326	187	14	0	0	0.39	0.13	0.00	0.00
	70	170	286	68	3655	937	2424	2003	709	0.87	0.32	0.19	0.03
	80	80	474	68	7117	1580	1481	4101	848	0.91	0.86	0.24	0.02
	90	175	n.a.	n.a.	10882	22216	53576	time	time	4.02	3.56	2.85	2.85
	100	54	n.a.	n.a.	18334	34389	33754	time	time	3.74	3.27	3.27	3.27
0.8	25	0	0	0	18	1	0	0	0	0.04	0.00	0.00	0.00
	30	0	0	0	26	2	1	0	0	0.02	0.01	0.00	0.00
	40	4	30	0	126	6	4	6	0	0.05	0.01	0.01	0.01
	50	0	0	0	253	21	0	0	0	0.11	0.00	0.00	0.00
	60	0	0	0	421	25	0	0	0	0.09	0.00	0.00	0.00
	70	4	194	0	1260	138	34	258	0	0.38	0.05	0.03	0.03
	80	30	168	22	3932	357	237	473	101	0.68	0.27	0.27	0.02
	90	38	6100	2228	5123	452	783	51097	26281	1.05	0.62	0.62	0.62
	100	98	9816	1260	8996	734	1566	64718	12828	1.04	0.61	0.61	0.61

Table 4: Detailed results of exact algorithm using CAB instances for PO_2

nine out of the 27 instances are optimally solved without any branching, including the 60 nodes instances for $\alpha = 0.5, 0.8$. Moreover, 15 instances are optimally solved in less than an hour of computing time. For only six instances the optimality of the best-known solution could not be proven within the time limit of one day.

8. Conclusions

In this paper we have proposed an exact algorithmic framework for hub network design problems with profits. In contrast to classical hub location problems, this class of problems do not assume all demand will be served and thus, the nodes that will be served and the commodities to be routed, must also be decided. We have considered two variants, which differ from each other in only one set of constraints that forces to route all the commodities with their two end-nodes activated. We proposed a Lagrangean relaxation that exploit the structure of the problems and can be solved efficiently. In particular, the Lagrangean functions can be decomposed in two independent subproblems: one of them is trivial and the other one can transformed into a Quadratic Boolean Problem, which can be solved efficiently as a max-flow problem. The Lagrangean dual problems were solved with a subgradient optimization algorithms that applied simple primal heuristics, which produced valid lower bounds. The Lagrangean relaxation was embedded within exact branch-and-bound algorithms for each of the considered problems. Moreover, reduction tests were applied at the root node, which helped to considerably reduce the number of variables and constraints. These tests were enhanced with the application of a partial enumeration phase to reduce the number of branches of the enumeration phase. The results from computational experiments with benchmark instances with up to 100 nodes assessed the efficiency of the proposed framework, and its superiority over CPLEX. On the one hand, because of memory limitations CPLEX was not able to solve instances with more than 60 or 70 nodes, depending on the version of the problem, whereas our proposed solution algorithms did not have this limitation. On the other hand, for the instances where both type of methods could be compared, our algorithms consistently outperformed CPLEX.

Acknowledgments

The research of the first two authors was partially funded by the Canadian Natural Sciences and Engineering Research Council [Grant 418609-2012]. The research of the third author was partially funded by the Spanish Ministry of Economy and Competitiveness and ERDF funds [Grant MTM2015-63779-R (MINECO/FEDER)]. This support is gratefully acknowledged.

References

Adler, N. and K. Smilowitz (2007). Hub-and-spoke network alliances and mergers: Price-location competition in the airline industry. *Transportation Research Part B 41*(4), 394–409.

Alibeyg, A., I. Contreras, and E. Fernández (2016). Hub network design problems with profits. *Transportation Research Part E: Logistics and Transportation Review*. DOI 10.1016/j.tre.2016.09.008.

Aykin, T. (1995). Networking policies for hub-and-spoke systems with application to the air transportation system. *Transportation Science 29*(3), 201–221.

Bryan, D. L. and M. E. O'Kelly (1999). Hub-and-spoke networks in air transportation: An analytical review. *Journal of Regional Science 39*(2), 275–295.

Campbell, J. F. (1994). Integer programming formulations of discrete hub location problems. *European Journal of Operational Research 72*(2), 387–405.

Campbell, J. F., A. Ernst, and M. Krishnamoorthy (2005). Hub arc location problems part I: Introduction and results. *Management Science 51*(10), 1540–1555.

Campbell, J. F. and M. E. O'Kelly (2012). Twenty-five years of hub location research. *Transportation Science 46*(2), 153–169.

Contreras, I. (2015). Hub location problems. In G. Laporte, F. Saldanha da Gama, and S. Nickel (Eds.), *Location Science*, pp. 311–344. Springer.

Contreras, I., J.-F. Cordeau, and G. Laporte (2011a). The dynamic uncapacitated hub location problem. *Transportation Science 45*(1), 18–32.

Contreras, I., J. A. Díaz, and E. Fernández (2011b). Branch and price for large-scale capacitated hub location problems with single assignment. *INFORMS Journal on Computing 23*(1), 41–55.

Contreras, I. and E. Fernández (2014). Hub location as the minimization of a supermodular set function. *Operations Research 62*(3), 557–570.

Contreras, I., E. Fernández, and A. Marín (2010). The tree of hubs location problem. *European Journal of Operational Research 202*(2), 390–400.

Contreras, I., M. Tanash, and N. Vidyarthi (2016). Exact and heuristic approaches for the cycle hub location problem. *Annals of Operations Research*. DOI 10.1007/s10479-015-2091-2.

de Camargo, R. S., G. Miranda, and H. P. Luna (2008). Benders decomposition for the uncapacitated multiple allocation hub location problem. *Computers and Operations Research 35*(4), 1047–1064.

Eiselt, H. A. and V. Marianov (2009). A conditional p-hub location problem with attraction functions. *Computers and Operations Research 36*(12), 3128–3135.

Elhedhli, S. and H. Wu (2010). A lagrangean heuristic for hub-and-spoke system design with capacity selection and congestion. *INFORMS Journal on Computing 22*(2), 282–296.

Gelareh, S., R. N. Monemi, and S. Nickel (2015). Multi-period hub location problems in transportation. *Transportation Research Part E: Logistics and Transportation Review 75*, 67–94.

Gelareh, S., S. Nickel, and D. Pisinger (2010). Liner shipping hub network design in a competitive environment. *Transportation Research Part E 46*(6), 991–1004.

Hwang, Y. H. and Y. H. Lee (2012). Uncapacitated single allocation p-hub maximal covering problem. *Computers and Industrial Engineering 63*(2), 382–389.

Jaillet, P., G. Song, and G. Yu (1996). Airline network design and hub location problems. *Location science 4*(3), 195–212.

Labbé, M. and H. Yaman (2008). Solving the hub location problem in a star–star network. *Networks 51*(1), 19–33.

Lin, C.-C. and S.-C. Lee (2010). The competition game on hub network design. *Transportation Research Part B 44*(4), 618–629.

Lowe, T. and T. Sim (2012). The hub covering flow problem. *Journal of the Operational Research Society 64*(7), 973–981.

Lüer-Villagra, A. and V. Marianov (2013). A competitive hub location and pricing problem. *European Journal of Operational Research 231*(3), 734–744.

Mahmutogullari, A. I. and B. Y. Kara (2016). Hub location under competition. *European Journal of Operational Research 250*(1), 214–225.

Marianov, V., D. Serra, and C. ReVelle (1999). Location of hubs in a competitive environment. *European Journal of Operational Research 114*(2), 363–371.

Martins de Sá, E., I. Contreras, and J.-F. Cordeau (2015a). Exact and heuristic algorithms for the design of hub networks with multiple lines. *European Journal of Operational Research 246*(1), 186–198.

Martins de Sá, E., I. Contreras, J.-F. Cordeau, R. Saraiva de Camargo, and G. de Miranda (2015b). The hub line location problem. *Transportation Science 49*(3), 500–518.

O'Kelly, M. E. (2012). Fuel burn and environmental implications of airline hub networks. *Transportation Research Part D: Transport and Environment 17*(7), 555–567.

O'Kelly, M. E., H. P. L. Luna, R. S. De Camargo, and G. de Miranda Jr (2015). Hub location problems with price sensitive demands. *Networks and Spatial Economics 15*(4), 917–945.

Orlin, J. B. (2012). Max flows in O(nm) time or better. In *Proceedings of the 2013 Symposium on the Theory of Computing*, pp. 765–774.

Picard, J.-C. and H. D. Ratliff (1975). Minimum cuts and related problems. *Networks 5*(4), 357–370.

Pirkul, H. and D. A. Schilling (1998). An efficient procedure for designing single allocation hub and spoke systems. *Management Science 44*(12), S235–S242.

Saberi, M. and H. S. Mahmassani (2013). Modeling the airline hub location and optimal market problems with continuous approximation techniques. *Journal of Transport Geography 30*, 68–76.

Sasaki, M., J. F. Campbell, A. Ernst, and M. Krishnamoorthy (2014). A Stackelberg hub arc location model for a competitive environment. *Computers and Operations Research 47*, 27–41.

Sasaki, M., A. Suzuki, and Z. Drezner (1999). On the selection of hub airports for an airline hub-and-spoke system. *Computers and Operations Research 26*(14), 1411–1422.

Algorithm 4 Exact Algorithm for PO_1: Branch and Bound Phase

```
Require: 3: (Branch on location variables $z$)
  Order $i \in H \setminus (H^0 \cup H^1)$ in non-increasing order with respect to $\delta_i$
  Branch on $z_i$ using the previous order and solve $D_1$ with Algorithm 1
  Let $AN^z$ be the set of remaining active nodes after fixing all $z_i$ variables
  if $UN^z \neq \emptyset$ then
      Go to Step 4
  else
      Stop with an $\epsilon$-optimal solution
  end if
Require: 4: (Branch on service variables $s$)
  for $a \in AN^z$ do
      Branch on $s_i$ variables for $i \in N \setminus H^1$ using an arbitrary order and
      solve $D_1$ with Algorithm 1
  end for
  Let $AN^s$ be the set of remaining active nodes after fixing all $s_i$ variables
  if $AN^s \neq \emptyset$ then
      Go to Step 5
  else
      Stop with an $\epsilon$-optimal solution
  end if
Require: 5: (Branch on hub-edge variables $z$)
  for $a \in AN^s$ do
      Branch on $y_e$ variables for $e \in H^1 \times H^1$ using an arbitrary order and
      solve $D_1$ with Algorithm 1
  end for
  Let $AN^y$ be the set of remaining active nodes after fixing all $y_e$ variables
  if $AN^y \neq \emptyset$ then
      Go to Step 6
  else
      Stop with an $\epsilon$-optimal solution
  end if
Require: 6: (Solve routing subproblems)
  for $a \in AN^y$ do
      Solve routing subproblems to obtain optimal values for $x_{ijk}$ variables
      and a valid lower bound $\eta_a$
      Update $\underline{\eta}$ if $\eta_a > \underline{\eta}$
  end for
```