Branch-and-Price for Large-Scale Capacitated Hub Location Problems with Single Assignment

Ivan Contreras[1], Juan A. Díaz[2], Elena Fernández[1]

[1] Dpt. d' Estadística i Investigació Operativa, Universitat Politècnica de Catalunya, 08034, Barcelona, Spain, {ivan.contreras@upc.edu, e.fernandez@upc.edu}

[2]Dpt. Ingeniería Industrial y Mecánica, Universidad de las Américas, 72820, Puebla, México, juana.diaz@udlap.mx

This paper presents a branch–and–price algorithm for the Capacitated Hub Location Problem with Single Assignment, in which Lagrangean relaxation is used to obtain tight lower bounds of the Restricted Master Problem (RPM). It is shown how to solve the pricing problem for finding new columns to be added to the RMP, or for proving that the lower bound of the current RMP is valid. Also, a lower bound that is valid at any stage of the column generation algorithm is proposed. The process to obtain this valid lower bound allows to apply a stabilization method that results in a considerable improvement on the overall efficiency of the solution algorithm. Numerical results on a battery of benchmark instances of up to 200 nodes are reported. These seem to be the largest instances that have been solved exactly for this problem.

Key words: Hub location; Lagrangean relaxation; Column generation.

---

1. Introduction

In Hub Location Problems (HLPs) customers send and receive some commodity that must be routed through a network. Hubs can be seen as transshipment points where the flows between origin-destination pairs are consolidated and re-routed to their destinations, sometimes via another hub. Two assumptions characterize most HLPs: *i*) All flows must be consolidated by hubs, so that the paths between origin-destination pairs must include at least one hub node; and, *ii*) Hubs are fully interconnected with more effective links that allow to apply a discount factor α, $0 < \alpha < 1$, to the costs of the flows routed between pairs of hubs. In HLPs it must be decided where to locate the hubs and how to route the flows between the origin-destination pairs via the selected hubs so as to minimize the total cost of the system, subject to the above assumptions. Many applications of HLPs arise in telecommunications, transportation, and postal delivery systems. The reader is addressed to Alumur and Kara [1] and Campbell et al. [7] for recent comprehensive surveys on hub location.

Most of the literature on HLPs deals with uncapacitated versions were it is assumed that there exists an unlimited capacity on the incoming and outgoing flow on the hubs, as well as on the amount of flow that transit through the connections between hubs. More realistic Hub Location models consider capacity constraints on the incoming and/or outgoing flow on the hubs and/or on the amount of flow that transits through the connections between hubs. Capacitated versions of HLPs with multiple assignment, where the demand of the non-hub nodes can be served from more than one hub have been considered, for instance, in Ebery et al. [10], Sasaki and Fukushima [18], Boland et al. [5], Marín [15] and Rodríguez-Martín and Salazar-González [17].

There are just a few papers in the literature dealing with Capacitated Hub Location Problems with Single Assignment (CHLPSA). The single assignment requirement imposes that for each non-hub node all the flow with origin/destination in the node must be sent/received from the same hub node. One application of the CHLPSA is described in [11], and corresponds to postal delivery where mail from postal districts has to be collected by a single mail sorting center. Campbell [6] considered capacity constraints on the incoming flow on the hubs coming from both non-hub and hub nodes, and proposed the first linear integer formulation with variables with four indices. Aykin [2] considered HLPs with capacity constraints on the incoming flow on the hubs as well as on direct origin-destination links. In [3] Aykin studied a similar problem where the number of hubs to locate is given. Ernst and Krishnamoorthy [11] proposed a formulation involving variables with three indices for the CHLPSA. Their branch-and-bound procedure that was able to solve instances up to 50 nodes. Labbé et al. [13] study several capacitated versions of HLPs with single assignment where the capacity constraints are on the incoming and/or outgoing flow on the hubs coming from both non-hub and/or hub nodes, depending the considered version. They present a formulation with variables of two indices and study some polyhedral properties. Their branch-and-cut method is able to solve instances up to 50 nodes. Costa et al. [9] deal with the capacitated version using a bi-criteria approach. Recently, Contreras, Díaz and Fernández [8] obtained tight lower bounds with a Lagrangean Relaxation based on a formulation with four-indices variables.

One of the main difficulties for solving HLPs, and in particular the CHLPSA, is the huge number of variables and constraints that typically appear in the considered models. From this point of view, formulations with fewer variables and constraints should be preferred to bigger formulations. However, most often bigger formulations lead to tighter bounds associated with their relaxations. In [11] it is seen that the lower bound from the LP relaxation of a CHLPSA formulation with four-indices variables is tighter than the one obtained using a formulation with three-indices variables, although at a considerable increase on the required cpu times. In [8] the Lagrangean

relaxation of the four-indices formulation was solved considerably faster than the LP relaxations of the formulations with both three and four indices, for instances up to 200 nodes. However, as the sizes of the instances increased the computation requirements of the Lagrangean relaxation again become too high. In addition, even if the percent gaps obtained at termination were very small, in most of the cases the optimality of the obtained solutions could not be proven.

Column Generation (CG) is a well-known decomposition method that has been successfully applied for obtaining tight bounds of huge linear and linear-integer programming problems. CG is typically used when a problem has too many variables so as to be solved explicitly. For a recent review of CG techniques and applications see, for instance, Lübbecke and Desrosiers [14] or Barnhart et al. [4]. The results of [8] suggest that column generation can be suitable for addressing the formulation of CHLPSA based on four-indices variables. This is the approach that we propose in this paper where, in addition, we use Lagrangean relaxation for efficiently obtaining a bound of the Restricted Master Problem at each global iteration of the solution algorithm. As we will see, the pricing problem can be solved efficiently by means of a series of semi-assignment problems.

It is well-known, that in CG the relaxation of the Restricted Master Problem (RPM) does not necessary yield a valid lower bound for the original problem. Thus, within the context of CG, valid lower bounds can be very useful since they can be used for extending the termination criteria when the gap between the value of the relaxation of the current RPM and the lower bounds are sufficiently small, or for applying variable elimination texts. In this work we propose two different lower bounds for CHLPSA that can be easily obtained. The process that produces the dual solutions associated with the second lower bound plays an important role in the overall efficiency of the CG algorithm, and can be seen as a stabilization mechanism, since by making use of these dual solutions, we obtain a much faster convergence of the method. We also propose elimination tests that allow us to fix or discard some of the hubs in optimal solutions. As could be expected, the reduction on the number of the variables of the problem has a beneficial effect on the efficiency of the overall algorithm.

Our column generation algorithm terminates with a valid lower bound, but not necessarily with an optimal solution to CHLPSA. Therefore, we propose a branch–and–price algorithm that guarantees the optimality of the best solution found. The enumeration tree consists of three phases: the first one is a partial enumeration on the hubs that have not been fixed so far. The second phase is an enumeration on the variables associated with hub location, and the third phase is an enumeration on the variables associated with the assignment variables.

To assess the efficiency of our algorithm we have run a series of computational experiments on

a set of benchmark instances from the literature with sizes up to 200 nodes, and we have optimally solved all the considered instances.

The rest of this work is structured as follows. Section 2 introduces formulation of the problem that we use, and recalls some previous results from [8]. In Section 3 we present the CG algorithm. The Restricted Master Problem is given in 3.1 and in 3.2 we describe how to solve its Lagrangean dual. The pricing problem is presented in 3.3, together with one of the two lower bounds and the overall CG algorithm. Section 4 describes the process to obtain the second lower bound and indicates how to use the associated dual solution for reaching a faster convergence of the CG algorithm. The reduction tests are given in Section 5 and the branch–and–price algorithm is presented in Section 6. The computational experiments that we have run together to the obtained numerical results are presented in Section 7. The paper concludes in Section 8 with some conclusions and comments on future research.

2. Mathematical Formulation

Let $G = (N, A)$ be a complete di-graph with set of nodes $N = \{1, 2, \cdots, n\}$, that correspond to origin/destinations. Let W_{ij} denote the flow emanating from node i with destination in node j, $O_i = \sum_{j \in N} W_{ij}$ the outgoing flow generated in node $i \in N$, and $D = \sum_{i \in N} O_i$ the total flow generated in the graph. The distance between nodes i and j is denoted d_{ij}, and we assume it satisfies the triangle inequality. Each node is a potential hub location, so that for each $k \in N$ it must be decided whether or not a hub will be located at k. Each non-hub node $i \in N$ must be assigned (allocated) to one single hub $k \in N$ in such a way that all the outgoing/incoming flow from/to i is sent through node k. Hub nodes are allocated to themselves. Thus, paths between origin/destination pairs are of the form $i - k - m - j$, where i and j represent the origin and destination, respectively; and k and m the hubs to which i and j are allocated, respectively. For evaluating the transportation costs of the per unit flow along the links of the graph, we use the distances of the links weighted by discount factors, denoted χ, α and δ, that depend on whether the links are used for collecting, transferring or distributing flow, respectively. Thus, the cost of routing one unit of flow along the path $i - k - m - j$ is given by $F_{ijkm} = W_{ij}\,(\chi d_{ik} + \alpha d_{km} + \delta d_{mj})$. For each $k \in N$, b_k and f_k, respectively denote the capacity and the fixed set-up cost of a hub at node k. The capacity of a hub is an upper bound on the total incoming flow that can be processed in the hub. Thus, it affects to the sum of the flow generated at the nodes assigned to the hub.

The CHLPSA consists of selecting a set of hubs to be established and an allocation pattern that

fully assigns each node to one of the chosen hubs, that does not violate the capacity constraint of the hubs, of minimum total cost.

In our formulation we define two sets of binary decision variables. For each pair $i, k \in N$

$$z_{ik} = \begin{cases} 1 & \text{if node } i \text{ is assigned to hub } k; \\ 0 & \text{otherwise.} \end{cases}$$

When $i = k$, variable z_{kk} represents the establishment or not of a hub at node k. The z variables will be referred to as location/allocation variables. We define one additional set of variables to represent the path used for sending the flow between each pair of nodes. For each $i, j, k, m \in N$ let

$$x_{ijkm} = \begin{cases} 1 & \text{if flow from } i \text{ to } j \text{ goes via hubs } k \text{ and } m; \\ 0 & \text{otherwise.} \end{cases}$$

Variables x will be referred to as routing variables. Our formulation of CHLPSA is:

$$(MP) \quad \min \quad \sum_{k\in N} f_k z_{kk} + \sum_{i\in N}\sum_{j\in N}\sum_{k\in N}\sum_{m\in N} F_{ijkm} x_{ijkm} \tag{1}$$

$$\text{s.t.} \quad \sum_{k\in N}\sum_{m\in N} x_{ijkm} = 1 \qquad \forall\, i, j \in N \tag{2}$$

$$z_{ik} \le z_{kk} \qquad \forall\, i, k \in N \tag{3}$$

$$\sum_{m\in N} x_{ijkm} = z_{ik} \qquad \forall\, i, j, k \in N \tag{4}$$

$$\sum_{k\in N} x_{ijkm} = z_{jm} \qquad \forall\, i, j, m \in N \tag{5}$$

$$\sum_{i\in N} O_i z_{ik} \le b_k z_{kk} \qquad \forall\, k \in N \tag{6}$$

$$\sum_{k\in N} b_k z_{kk} \ge D \tag{7}$$

$$x_{ijkm} \ge 0 \qquad \forall\, i, j, k, m \in N \tag{8}$$

$$z_{ik} \in \{0, 1\} \qquad \forall\, i, k \in N \tag{9}$$

Constraints (2) guarantee that for each pair of nodes there is one single path connecting them, whereas constraints (3) impose that no customer is assigned to a node that is not a hub. Constraints (4) state that if node i is assigned to hub k then all the flow from node i to any other (fixed) node j must go through some other hub m. Constraints (5) have a similar interpretation relative to the flow arriving to a node j assigned to hub m from some node i. Note that constraints (2) together with constraints (4) and (5) ensure that every node is assigned to one single hub. In addition, given that the z variables are binary, they guarantee the integrality of the x variables. Constraints (6) ensure that the overall incoming flow of nodes assigned to a hub does not exceed its capacity.

Constraint (7) is the aggregated demand constraint. This constraint is redundant in model M1, since it can be derived by adding up all constraints (6), and taking into account equalities (2) and (4). We include it in the formulation in order to strengthen a Lagrangean relaxation that we will consider.

A model similar to MP but without the aggregated demand constraint (7) was proposed by Campbell [6], and was computationally tested by Ernst and Krishnamoorthy in [11]. Their computational experiments showed that the formulation that used variables with four indices lead to tighter LP bounds than the one obtained with a formulation that used variables with three indices, at a considerable increase on the required cputimes. In [8] Contreras, Díaz and Fernández used formulation MP and proposed a Lagrangean Relaxation associated with constraints (4) and (5). In particular, weighting constraints (4) and (5) in a Lagrangean fashion, with multipliers vectors u and v of appropriate dimensions, we obtain a Lagrangean function which, after some algebra (see [8] for details), can be expressed as $L(u,v) = L_z(u,v) + L_x(u,v)$, where

$$
\begin{aligned}
L_z(u,v) = \min \quad & \sum_{k\in N} f_k z_{kk} - \sum_{i\in N}\sum_{k\in N}\left(\sum_{j\in N}(u_{ijk}+v_{jik})\right) z_{ik} && (10)\\
\text{s.t.} \quad & z_{ik} \le z_{kk} \qquad \forall\, i,k \in N && (11)\\
& \sum_{i\in N} O_i z_{ik} \le b_k z_{kk} \qquad \forall\, k \in N && (12)\\
& \sum_{k\in N} b_k z_{kk} \ge D && (13)\\
& z_{ik} \in \{0,1\} \qquad \forall\, i,k \in N, && (14)
\end{aligned}
$$

and

$$
\begin{aligned}
L_x(u,v) = \min \quad & \sum_{i\in N}\sum_{j\in N}\sum_{k\in N}\sum_{m\in N}(F_{ijkm}+u_{ijk}+v_{ijm})\, x_{ijkm} && \\
\text{s.t.} \quad & \sum_{k\in N}\sum_{m\in N} x_{ijkm} = 1 \qquad \forall\, i,j \in N && (15)\\
& x_{ijkm} \ge 0 \qquad \forall\, i,j \in N, \forall\, k,m \in N. && (16)
\end{aligned}
$$

The following two propositions were proven in [8].

Proposition 1 *The optimal solution to $L_z(u,v)$ can be obtained by solving the problem:*

$$L_z(u,v) = min \quad \sum_{k\in N}\left(f_k - \sum_{j\in N}(u_{kjk}+v_{jkk}) - \xi_k(u,v)\right) z_{kk}$$

$$s.t. \quad \sum_{k\in N} b_k z_{kk} \geq D \tag{17}$$

$$z_{kk} \in \{0,1\} \qquad \forall\, k \in N, \tag{18}$$

$$where \quad \xi_k(u,v) = min \quad \sum_{i\in N: i\neq k}\left(\sum_{j\in N}(u_{ijk}+v_{jik})\right) z_{ik}$$

$$s.t. \quad \sum_{i\in N: i\neq k} O_i z_{ik} \leq (b_k - O_k) \qquad \forall\, k \in N \tag{19}$$

$$z_{ik} \in \{0,1\} \qquad \forall\, i,k \in N, i \neq k. \tag{20}$$

Thus, we can solve $L_z(u,v)$ with a series of $|N|+1$ knapsack problems. Although knapsack problems are *NP-Hard*, they can be solved very efficiently (see, for instance [16]).

Proposition 2 *Subproblem $L_x(u,v)$ can be can be expressed as the sum of $|A|$ independent semi-assignment problems of the form:*

$$(SAP_{ij}) \quad q_{ij}(u,v) = min \quad \sum_{k\in N}\sum_{m\in N}(F_{ijkm} + u_{ijk} + v_{ijm})\, x_{ijkm}$$

$$s.t. \quad \sum_{k\in N}\sum_{m\in N} x_{ijkm} = 1 \tag{21}$$

$$x_{ijkm} \geq 0 \qquad \forall\, k,m \in N. \tag{22}$$

For a given (i,j) pair, the optimal value to SAP_{ij} is $q_{ij}(u,v) = F_{ij\widehat{k}\widehat{m}} + u_{ij\widehat{k}} + v_{ij\widehat{m}}$, where $\widehat{k},\widehat{m} \in N$ are arbitrarily selected such that

$$F_{ij\widehat{k}\widehat{m}} + u_{ij\widehat{k}} + v_{ij\widehat{m}} = min\ \{F_{ijkm} + u_{ijk} + v_{ijm} : k,m \in N\}. \tag{23}$$

Therefore, $L_x(u,v) = \sum_{i,j\in N} q_{ij}(u,v) = \sum_{i,j\in N} min\ \{F_{ijkm} + u_{ijk} + v_{ijm} : k,m \in N\}$.

Hence, we can obtain the solution to $L_x(u,v)$ by solving a series of $|A|$ semi-assignment subproblems. For a given pair $i,j \in N$, evaluating the closed expression (23) has a complexity of $\mathcal{O}(|N|^2)$. Thus, solving all the semi-assignment problems is $\mathcal{O}(|N|^4)$.

A lower bound for MP is z_D, the optimal value of the Lagrangean dual problem

$$(D) \quad z_D = \max_{u,v} \quad L(u,v). \tag{24}$$

Problem D can be solved with subgradient optimization. For a given vector (u,v), let $z(u,v)$, and $x(u,v)$ denote the optimal solution to $L(u,v)$. Then, a subgradient of $L(u,v)$ is:

$$\gamma(u,v) = \left(\left(\sum_{m \in N} x_{ijkm}(u,v) - z_{ik}(u,v) \right)_{i,j,k \in N}, \quad \left(\sum_{k \in N} x_{ijkm}(u,v) - z_{jm}(u,v) \right)_{i,j,m \in N} \right).$$

3. Column Generation approach for the CHLPSA

As pointed out above, the results of [8] indicate that the Lagrangean dual (24) produces very tight lower bounds for MP, and it is considerably less time consuming than the LP relaxation of MP. However the computation requirements for solving D become too high as the size of the instances increase. Thus, column generation can be suitable for addressing formulation MP and for efficiently obtaining these tights bounds. This is the methodology that we apply next for solving MP. In the following subsections we describe the different elements of our column generation algorithm.

3.1 The Restricted Master Problem

We consider a restriction of MP that uses all the columns associated with allocation/location variables z, but only a subset of the columns associated with the routing variables x. For each pair $i, j \in N$, we restrict the set of *potential hubs* for sending flow from i to j to a subset $S_{ij} \subseteq N$. Let $S = \{S_{ij}\}_{i,j \in N}$ denote the set of all subsets of potential hubs (that, abusing slightly notation, we also denote *set of potential hubs*). Also, let $X(S) = \{x_{ijkm} : i, j \in N; k, m \in S_{ij}\}$ denote the subset of columns of the x variables induced by S. The Restricted Master Problem (RMP) is:

$$(RMP) \quad z(S) = \min \quad \sum_{k \in N} f_k z_{kk} + \sum_{i \in N} \sum_{j \in N} \sum_{k \in S_{ij}} \sum_{m \in S_{ij}} F_{ijkm} x_{ijkm} \tag{25}$$

$$\text{s.t.} \quad \sum_{k \in S_{ij}} \sum_{m \in S_{ij}} x_{ijkm} = 1 \qquad \forall\, i, j \in N \tag{26}$$

$$z_{ik} \leq z_{kk} \qquad \forall\, i, k \in N \tag{27}$$

$$\sum_{m \in S_{ij}} x_{ijkm} = z_{ik} \qquad \forall\, i, j, k \in N \tag{28}$$

$$\sum_{k \in S_{ij}} x_{ijkm} = z_{jm} \qquad \forall\, i, j, m \in N \tag{29}$$

$$\sum_{i \in N} O_i z_{ik} \leq b_k z_{kk} \qquad \forall\, k \in N \tag{30}$$

$$\sum_{k \in N} b_k z_{kk} \geq D \tag{31}$$

$$x_{ijkm} \geq 0 \qquad \forall\, i, j \in N, \forall\, k, m \in S_{ij} \tag{32}$$

$$z_{ik} \in \{0, 1\} \qquad \forall\, i, k \in N \tag{33}$$

3.2 Lagrangean relaxation of the Restricted Master Problem

It is clear that the linear programming relaxation of RMP can be used for obtaining a lower bound for RPM. However, we can also use a stronger relaxation like, for instance, a Lagrangean relaxation. This is the approach that is followed here where we solve the Lagrangean relaxation of RPM described in Section 2. In particular, the Lagrangean function that we obtain when relaxing in a Lagrangean fashion constraints (28) and (29) of the RPM associated with the set of potential hubs S is $L^S(u,v) = L_z(u,v) + L_x^S(u,v)$, where $L_z(u,v)$ is exactly subproblem (10)-(14), since it does not depend on the subsets of potential hubs S. On the contrary, subproblem $L_x^S(u,v)$ depends on the sets S_{ij}, $i,j \in N$ and is given by

$$L_x^S(u,v) = \min \quad \sum_{i\in N}\sum_{j\in N}\sum_{k\in S_{ij}}\sum_{m\in S_{ij}} (F_{ijkm} + u_{ijk} + v_{ijm})\, x_{ijkm}$$

$$\text{s.t.} \quad \sum_{k\in S_{ij}}\sum_{m\in S_{ij}} x_{ijkm} = 1 \qquad \forall\, i,j \in N \tag{34}$$

$$x_{ijkm} \geq 0 \qquad \forall\, i,j \in N, \forall\, k,m \in S_{ij}. \tag{35}$$

Similarly to $L_x(u,v)$, $L_x^S(u,v)$ can be decomposed into $|A|$ semi-assignment problems of the form

$$(SAP_{ij}^{S_{ij}}) \quad q_{ij}^S(u,v) = \min \quad \sum_{k\in S_{ij}}\sum_{m\in S_{ij}} (F_{ijkm} + u_{ijk} + v_{ijm})\, x_{ijkm}$$

$$\text{s.t.} \quad \sum_{k\in S_{ij}}\sum_{m\in S_{ij}} x_{ijkm} = 1 \tag{36}$$

$$x_{ijkm} \geq 0 \qquad \forall\, k,m \in S_{ij}. \tag{37}$$

Subproblem $SAP_{ij}^{S_{ij}}$ is similar to subproblem SAP_{ij} with the only difference that the variables of $SAP_{ij}^{S_{ij}}$ are restricted to the subset S_{ij}, whereas SAP_{ij} uses all variables x_{ijkm}, $k,m \in N$. Thus, $\forall i,j \in N$, the value $q_{i,j}^S(u,v)$ can be obtained with a closed expression similar to (23), where $\widehat{k},\widehat{m} \in S_{ij}$ are arbitrarily selected as in (23) but restricted to the set S_{ij}. That is,

$$q_{ij}^S(u,v) = F_{ij\widehat{k}\widehat{m}} + u_{ij\widehat{k}} + v_{ij\widehat{m}} = \min\, \{F_{ijkm} + u_{ijk} + v_{ijm} : k,m \in S_{ij}\}\,. \tag{38}$$

Therefore, $L_x^S(u,v) = \sum\limits_{i,j\in N} q_{ij}^S(u,v) = \sum\limits_{i,j\in N} \min\, \{F_{ijkm} + u_{ijk} + v_{ijm} : k,m \in S_{ij}\}$.

For a given pair $i,j \in N$, evaluating the closed expression (38) has a complexity of $\mathcal{O}(|S_{ij}|^2)$. Thus, in the worst case, solving all the semi-assignment problems is $\mathcal{O}(|S|^4)$.

Remark 1 Since subproblem $SAP_{ij}^{S_{ij}}$ is a linear programming problem, the optimal dual variable $w_{ij}^S(u,v)$ associated with the corresponding constraint (26) can be obtained from the well-known linear programming expression $c_B B^{-1}$, where B is the coefficient matrix of the optimal basis and c_B

the cost vector of basic variables. Given that $SAP_{ij}^{S_{ij}}$ has one single constraint and all the coefficients in the left hand side are ones, we have $B = B^{-1} = 1$. Thus, $w_{ij}^S(u,v) = c_B = F_{ij\widehat{k}\widehat{m}} + u_{ij\widehat{k}} + v_{ij\widehat{m}}$, where $\widehat{k}, \widehat{m} \in S_{ij}$ are arbitrarily selected as in (38). That is, $\forall i,j \in N$, $w_{ij}^S(u,v) = q_{ij}^S(u,v)$, so that we can also write $L_x^S(u,v) = \sum_{i,j\in N} q_{ij}^S(u,v) = \sum_{i,j\in N} w_{ij}^S(u,v)$. Observe that, by definition of $w_{ij}^S(u,v)$, it holds that $-w_{ij}^S(u,v) + F_{ijkm} + u_{ijk} + v_{ijm} \geq 0$, $\forall i,j \in N, \forall k,m \in S_{ij}$.

A lower bound for the RMP associated with the set of potential hubs S is z_D^S, the optimal value of the Lagrangean dual problem

$$(D^S) \quad z_D^S = \max_{u,v} \quad L^S(u,v). \tag{39}$$

Now, a subgradient of $L^S(u,v)$ is given by

$$\gamma^S(u,v) = \left(\left(\sum_{m\in S_{ij}} x_{ijkm}(u,v) - z_{ik}(u,v) \right)_{i,j,k\in N}, \quad \left(\sum_{k\in S_{ij}} x_{ijkm}(u,v) - z_{jm}(u,v) \right)_{i,j,m\in N} \right).$$

where $z(u,v)$, and $x(u,v)$ denote the optimal solution to $L^S(u,v)$.

We end this subsection by comparing the lower bounds of the RPMs associated with different sets of potential hubs.

Proposition 3 *Let S, S' be two subsets of potential hubs such that $S_{ij} \subseteq S'_{ij}$, $\forall i,j \in N$. Then, $z_D^S \geq z_D^{S'}$.*

Proof

For any vector u, v of appropriate dimensions we have

$\min\{F_{ijkm} + u_{ijk} + v_{ijm} : k,m \in S_{ij}\} \geq \min\{F_{ijkm} + u_{ijk} + v_{ijm} : k,m \in S'_{ij}\}$, $\forall i,j \in N$.

Hence, $q_{ij}^S(u,v) \geq q_{ij}^{S'}(u,v)$, $\forall i,j \in N$, so that $L_x^S(u,v) \geq L_x^{S'}(u,v)$.

Therefore, $L^S(u,v) = L_z(u,v) + L_x^S(u,v) \geq L_z(u,v) + L_x^{S'}(u,v) = L^{S'}(u,v)$.

As a consequence, $z_D^S = \max_{u,v} L_x^S \geq \max_{u,v} L_x^{S'} = z_D^{S'}$. ■

3.3 The Pricing Problem

While the optimal value z_D^S is a lower bound on the value of the RPM associated with S, it is well-known that z_D^S need not be a valid lower bound on the optimal value of MP. The following proposition indicates under which conditions z_D^S is a valid lower bound on MP.

Proposition 4 *Let $S = \{S_{ij}\}_{i,j\in N}$ be a given set of potential hubs. Let also (u^S, v^S) denote the optimal solution to D^S, and $w^S = (w^S_{ij})_{ij\in N} = (w^S_{ij}(u,v))_{i,j\in N} = (q^S_{ij}(u,v))_{i,j\in N}$. If $\bar{c}_{ijkm} = F_{ijkm} + u^S_{ijk} + v^S_{ijm} - w^S_{ij} \geq 0$, $\forall i,j,k,m \in N$, then the value z^S_D is a valid bound for MP.*

Proof

For a given $i,j \in N$, by definition of w^S_{ij} we have

$$\bar{c}_{ijkm} \geq 0 \Leftrightarrow F_{ijkm} + u^S_{ijk} + v^S_{ijm} \geq \min\{F_{ijk'm'} + u^S_{ijk'} + v^S_{ijm'} : k', m' \in S_{ij}\}.$$

Thus,

$$\begin{aligned} q_{ij}(u^S, v^S) &= \min\{F_{ijkm} + u^S_{ijk} + v^S_{ijm} : k,m \in N\} = \\ &= \min\{F_{ijkm} + u^S_{ijk} + v^S_{ijm} : k,m \in S_{ij}\} = q^S_{ij}(u^S, v^S), \end{aligned}$$

so that

$$L_x(u^S, v^S) = \sum_{i,j\in N} q_{ij}(u^S, v^S) = \sum_{i,j\in N} q^S_{ij}(u^S, v^S) = L^S_x(u^S, v^S).$$

Hence,

$$z_D \geq L(u^S, v^S) = L_z(u^S, v^S) + L_x(u^S, v^S) = L_z(u^S, v^S) + L^S_x(u^S, v^S) = L^S(u^S, v^S) = z^S_D \qquad \blacksquare$$

Let S be a set of potential hubs that generates a valid lower bound z^S_D. When applying Proposition 3 with sets of potential hubs $S'_{ij} = N$, $\forall i,j \in N$ we have:

Corollary 1 *Let $S = \{S_{ij}\}_{i,j\in N}$ be a given set of potential hubs. Let also (u^S, v^S) denote the optimal solution to D^S, and $w^S = (w^S_{ij})_{ij\in N} = (w^S_{ij}(u,v))_{i,j\in N} = (q^S_{ij}(u,v))_{i,j\in N}$. If $\bar{c}_{ijkm} = F_{ijkm} + u^S_{ijk} + v^S_{ijm} - w^S_{ij} \geq 0$, $\forall i,j,k,m \in N$, then $z_D = z^S_D$.*

The pricing problem is a test for validating the lower bound z^S_D associated with given RPM. It has a double effect. On the one hand, it checks whether or not the conditions of Proposition 4 hold for verifying if z^S_D is a valid lower bound. On the other hand, when the bound cannot be validated, it provides with new columns associated with new potential hubs that contribute to obtain a valid lower bound.

Therefore, a valid lower bound for the optimal value of the MP can be obtained by iteratively solving the relaxations of a series of RMPs. At each iteration we resort to the pricing problem to validate the lower bound or to find out new variables x_{ijkm}, not belonging to the current $X(S)$, which, if incorporated to the set S, could decrease the value of the lower bound of the associated RPM. The iterative process is repeated until no new columns are identified. The details of the pricing problem for the RPM associated with a given set of potential hubs S are as follows.

Let (u^S, v^S) denote the optimal solution to D^S, and let $w^S = (q_{ij}^S(u,v))_{i,j\in N}$. For each pair $i,j \in N$, the route $i-k-m-j$ from i to j with minimum $\bar{c}_{ijkm} = F_{ijkm} + u_{ijk}^S + v_{ijm}^S - w_{ij}^S$ can be determined from the solution to the semi-assignment problem:

$$\min \quad \sum_{k\in N}\sum_{m\in N} \bar{c}_{ijkm} x_{ijkm} \tag{40}$$

$$\text{s.t.} \quad \sum_{k\in N}\sum_{m\in N} x_{ijkm} = 1 \tag{41}$$

$$x_{ijkm} \geq 0 \qquad \forall\, k,m \in N, \tag{42}$$

which, in view of constraints (21), can be written as

$$-w_{ij}^S + q_{ij}(u^S, v^S) = -w_{ij}^S + \min \quad \sum_{k\in N}\sum_{m\in N} \left(F_{ijkm} + u_{ijk}^S + v_{ijm}^S\right) x_{ijkm}$$

$$\text{s.t.} \quad \sum_{k\in N}\sum_{m\in N} x_{ijkm} = 1$$

$$x_{ijkm} \geq 0 \qquad \forall\, k,m \in N,$$

Thus, the pricing problem reduces to solving a series of semi-assignment problems for obtaining $q_{ij}(u^S, v^S)$, $\forall i,j \in N$. Recall that the optimal value of each of these problems can be obtained by means of the closed expression (23). When $-w_{ij}^S + q_{ij}(u^S, v^S) \geq 0$, $\forall i,j \in N$, then the conditions of Corollary 1 hold, so that $z_D = z_D^S$ and z_D^S is a valid lower bound. Otherwise, for each $i,j \in N$ such that $-w_{ij}^S + q_{ij}(u^S, v^S) < 0$ we enlarge the set of potential hubs to $S_{ij} := S_{ij} \cup \{\widehat{k}, \widehat{m}\}\}$, with $\widehat{k}, \widehat{m}$ defined as in (23).

Even if the optimal value of the relaxation of RPM is not necessarily a valid lower bound for MP, after solving the pricing problem we can obtain a valid lower bound for MP as indicated in the following proposition.

Proposition 5 *For a given set of potential hubs S, let z_D^S denote the optimal value of the relaxation of the associated RPM. Then a valid lower bound for MP is given by*

$$LB^1(S) = z_D^S + \sum_{i,j\in N} min\ \left\{0, -w_{ij}^S + q_{ij}(u^S, v^S)\right\}.$$

Proof

The result follows since by Constraints (2), for each pair $i,j \in N$, there is exactly one variable x_{ijkm} at value one in any feasible solution to MP. ■

Note that the bound $LB^1(S)$ is immediately available after (u^S, v^S) is given. Note also, that valid lower bound $LB^1(S)$ can be obtained for any vector (u,v) (not necessarily (u^S, v^S)).

3.4 Overall CG Algorithm

Algorithm 3.4 describes the column generation procedure.

Algorithm 3 Column Generation Method

Input: Initial set of potential hubs S
StopCriterion $\leftarrow$ **false**
repeat
  StopCriterion $\leftarrow$ **true**
  Solve D^S for obtaining (u^S, v^S) and z_D^S.
  Evaluate w^S
  forall $(i, j \in N)$ **do**
    Identify $\widehat{k}$ and $\widehat{m}$, and evaluate $q_{ij}(u^S, v^S)$
    if $(-w_{ij}^S + q_{ij}(u^S, v^S) < 0)$ *then*
      Update $S_{ij} \leftarrow S_{ij} \cup \{\widehat{k}, \widehat{m}\}$
      StopCriterion $\leftarrow$ **false**
    end if
  enddo
until (*StopCriterion*)
Output: Valid lower bound $LB^0 = z_D^S$.

The output of Algorithm 3.4 returns a valid lower bound LB^0, which is the optimal value of the Lagrangean dual that is solved in the last iteration. Since the algorithm terminates with a subset of columns S and optimal dual variables (u^S, v^S), $w_{ij}^S = q_{ij}^S(u^S, v^S)$, such that $-w_{ij}^S + q_{ij}(u^S, v^S) = 0$, $\forall i, j \in N$, at termination we have $LB^1(S) = z_D^S$, so that $LB^0 = z_D^S = LB^1(S)$.

In our implementation of Algorithm 3.4 the initial set of columns is obtained with the GRASP heuristic described in the Appendix A.

4. Stabilization of CG

As indicated in Corollary 1 the lower bound obtained at termination of the column generation algorithm is the same than the lower bound that we obtain if we solve one single Lagrangean dual with unrestricted set of potential hubs, i.e. $S = N$. The only difference between the Lagrangean Dual with unrestricted set of potential hubs $S = N$ and the Lagrangean Duals with restricted set of potential hubs $S \subset N$ is the set of variables in the semi-assignment subproblems SAP_{ij}, $i, j \in N$. That is, when applying Algorithm 3.4, instead of solving one Lagrangean Dual with "large" semi-assignment subproblems SAP_{ij}, $i, j \in N$, we solve a series of Lagrangean Duals with smaller semi-assignment subproblems $SAP_{ij}^{S_{ij}}$, whose sizes increase from one Lagangean Dual to the next one. Even if each semi-assignment subproblem can be easily solved, the overall complexity of

$\mathcal{O}(|S|^4)$ for solving all the semi-assignment problems at each iteration of subgradient optimization can be quite time consuming for $S = N$. Therefore, it seems that solving a series of Lagrangean Duals of increasing sizes will be less time consuming than solving one single large Lagrangean Dual. However, this idea is not supported by the numerical results from the Computational Experiments Section, where we will see that the convergence of the basic version of Algorithm 3.4 is considerably slower than that of Lagrangean Relaxation with unrestricted set of hubs $S = N$.

There are two main weaknesses of the basic CG algorithm relative to Lagrangean Relaxation with unrestricted set of hubs $S = N$. The first one is that Lagrangean Relaxation with unrestricted set of hubs generates a valid lower bound at each iteration of subgradient optimization, whereas the basic CG does not. In the Lagrangean Relaxation with unrestricted set of hubs the lower bounds usually allow to eliminate many variables so that, in practice, the sizes of the semi-assignment subproblems that are actually solved with Lagrangean relaxation are not larger than those of CG. This weakness can be improved by incorporating valid lower bounds (for instance, $LB^1(S)$) to CG that also allow to apply elimination tests. The second weakness of the basic CG refers to the slow convergence of Algorithm 3.4. While for small size instances the number of global iterations is moderate (less than 10), for medium size and large instances the convergence of the column generation algorithm becomes very slow, and the efficiency of the overall algorithm is not good.

One possibility for speeding up the convergence of Algorithm 3.4 is to extend the termination criterion so as to allow termination when the gap between the lower bound for the current RPM, z_D^S, and the valid lower bound of Proposition 5, $LB^1(S)$, is below a given threshold value ϵ. Indeed, when $z_D^S - LB^1(S) < \epsilon$, even if additional iterations of the column generation algorithm may provide a lower bound better than $LB^1(S)$, the improvement on the lower bound may not compensate the additional computational burden. In fact, this extended termination criterion can be applied to any valid lower bound for MP, and the effect of the test on the overall performance of Algorithm 3.4 relies strongly on the quality of the lower bound. While lower bound $LB^1(S)$ is immediately available, in general, its quality is not very good. For this reason we propose a method to obtain, a lower bound for MP that is different from $LB^1(S)$. The new lower bound, $LB^2(S)$ can be obtained efficiently, although it requires some additional computations. Since $LB^2(S)$ yields tighter bounds than $LB^1(S)$, the extended termination criterion is applied more efficiently with $LB^2(S)$, resulting on an improved overall performance of the algorithm. In addition, and, more important, when the lower bound $LB^2(S)$ is obtained every fixed number of iterations of the subgradient optimization algorithm, the dual solutions associated with the new lower bound, contribute to generate a sequence of dual solutions that converge to the optimal Lagrangean dual

solution much faster than with the standard subgradient optimization algorithm.

The idea is to consider lower bounds $L(\widehat{u},\widehat{v})$ associated with vectors $(\widehat{u},\widehat{v})$ obtained by perturbing (u^S,v^S) so as to minimize the difference $z_D^S - L(\widehat{u},\widehat{v}) = L^S(u^S,v^S) - L(\widehat{u},\widehat{v})$.

If we set $\widehat{u} = u^S$, $\widehat{v} = v^S$ we have $L_z(\widehat{u},\widehat{v}) = L_z(u^S,v^S)$, and, as we have already seen, $L_x(\widehat{u},\widehat{v})$ will be different from $L_x^S(u^S,v^S)$ unless $L^S(u^S,v^S)$ already defines a valid lower bound for MP. Indeed this possibility leads to the already considered lower bound $L(u^S,v^S) = L_z(u^S,v^S) + L_x(u^S,v^S) = LB^1(S)$.

A different possibility, that we develop next, is to define $\widehat{u} = u^S+\alpha$ and $\widehat{v} = v^S+\beta$ in such a way that $L_x(\widehat{u},\widehat{v}) = L_x^S(u^S,v^S)$. Recall that, in general, there are two main differences between $L_x(\widehat{u},\widehat{v})$ and $L_x^S(u^S,v^S)$. On the one hand, the set of variables of L_x^S is a subset of the set of variables of L_x. On the other hand, in $L_x(\widehat{u},\widehat{v})$ the cost coefficients are $F_{ijkm} + \left(u_{ijk}^S + \alpha_{ijk}\right) + \left(v_{ijm}^S + \beta_{ijm}\right)$, whereas in $L_x^S(u^S,v^S)$ the cost coefficients are $F_{ijkm} + u_{ijk}^S + v_{ijm}^S$. One way of imposing that $L_x(\widehat{u},\widehat{v}) = L_x^S(u^S,v^S)$ is by forcing that:

i) The solution $x(\widehat{u},\widehat{v})$ that solves the Lagrangean function $L_x(\widehat{u},\widehat{v})$ is the same as the solution $x(u,v)$ that solves $L_x^S(u,v)$ (in the sense that the components at value one are the same in both solutions).

ii) The cost coefficients of the variables at value one are the same in $L_x(\widehat{u},\widehat{v})$ and in $L_x^S(u,v)$.

That is, $\alpha_{ij\widehat{k}} = \beta_{ij\widehat{m}} = 0$, $\forall i,j \in N$, where $\widehat{k},\widehat{m} \in S_{ij}$ are such that $x_{ij\widehat{k}\widehat{m}} = 1$ in the optimal solution to $SAP_{ij}^{S_{ij}}$.

Under condition *ii)*, condition *i)* can be stated as

$$q_{ij}(u+\alpha, v+\beta) = q_{ij}^S(u,v), \qquad \forall i,j \in N,$$

which, in turn, is equivalent to

$$F_{ijkm} + \left(u_{ijk}^S + \alpha_{ijk}\right) + \left(v_{ijm}^S + \beta_{ijm}\right) \geq q_{ij}^S(u,v), \quad \forall i,j,k,m \in N \text{ such that } k \neq \widehat{k} \text{ or } m \neq \widehat{m}.$$

Now, the lower bound is $LB^2(S) = L_z(\widehat{u},\widehat{v}) + L_x(u^S,v^S)$ that differs from z_D^S in $L_z(u^S,v^S) - L_z(\widehat{u},\widehat{v})$. By reducing the difference $L_z(u^S,v^S) - L_z(\widehat{u},\widehat{v})$ we generate valid lower bounds which are close to z_D^S. One way of reducing the difference $L_z(u^S,v^S) - L_z(\widehat{u},\widehat{v})$ is by minimizing $\sum\limits_{i,j\in N}\left(\sum\limits_{k\in N}\alpha_{ijk} + \sum\limits_{m\in N}\beta_{ijm}\right)$. Therefore, for finding a vector $(\widehat{u},\widehat{v})$, that produces a lower bound $LB^2(S) = L(\widehat{u},\widehat{v})$, with a small difference from z_D we formulate the following problem:

$$\min \quad \sum_{i\in N}\sum_{j\in N}\left(\sum_{k\in N\setminus\widehat{k}}\alpha_{ijk}+\sum_{m\in N\setminus\widehat{m}}\beta_{ijm}\right) \tag{43}$$

$$\text{s.t.} \quad F_{ijkm}+\left(u^S_{ijk}+\alpha_{ijk}\right)+\left(v^S_{ijm}+\beta_{ijm}\right)-q^S_{ij}(u^S,v^S)\geq 0 \qquad \forall\, i,j,k,m\in N \tag{44}$$

$$\alpha_{ijk}\geq 0 \qquad \forall i,j\in N, \forall k\in N\setminus\widehat{k} \tag{45}$$

$$\beta_{ijm}\geq 0 \qquad \forall i,j\in N, \forall m\in N\setminus\widehat{m} \tag{46}$$

which can be decomposed in $|A|$ subproblems of the form

$$(P^{ij}_{\alpha\beta}) \qquad \min \quad \sum_{k\in N\setminus\widehat{k}}\alpha_{ijk}+\sum_{m\in N\setminus\widehat{m}}\beta_{ijm} \tag{47}$$

$$\text{s.t.} \quad \alpha_{ijk}+\beta_{ijm}\geq \delta^{ij}_{km} \qquad \forall\, k,m\in N \tag{48}$$

$$\alpha_{ijk}\geq 0 \qquad \forall k\in N\setminus\widehat{k} \tag{49}$$

$$\beta_{ijm}\geq 0 \qquad \forall m\in N\setminus\widehat{m} \tag{50}$$

where $\delta^{ij}_{km}=\max\left\{0, \quad q^S_{ij}(u^S,v^S)-\left(F_{ijkm}+u^S_{ijk}+v^S_{ijm}\right)\right\}$, $\forall k,m\in N$.
Observe that the dual of subproblem $P^{ij}_{\alpha\beta}$ is

$$\max \quad \sum_{k,m\in N}\delta^{ij}_{km}\gamma_{km} \tag{51}$$

$$\text{s.t.} \quad \sum_{m\in N}\gamma_{km}\leq 1 \qquad \forall\, k\in N \tag{52}$$

$$\sum_{k\in N}\gamma_{km}\leq 1 \qquad \forall\, m\in N \tag{53}$$

$$\gamma_{km}\geq 0 \qquad \forall k,m\in N \text{ such that } k\neq\widehat{k} \text{ or } m\neq\widehat{m}. \tag{54}$$

which is a assignment problem with inequality constraints and, thus, can be solved in $\mathcal{O}(|S|^3)$, with the Hungarian algorithm [12]. In practice, the size of the assignment problems $P^{ij}_{\alpha\beta}$ can be reduced by means of a very simple preprocess. First, all constraints (48) with $\delta^{ij}_{km}\leq 0$ can be eliminated. Also, for each $k\in N$, such that $\delta^{ij}_{k\widehat{m}}>0$, we can set $\beta_{ijm}=\delta^{ij}_{k\widehat{m}}>0$, and, for this k update correspondingly the δ^{ij}_{km}'s for $m\in N$. Similarly, for each $m\in N$, such that $\delta_{\widehat{k}^{ij}m}>0$, we can set $\alpha_{ijk}=\delta^{ij}_{\widehat{k}m}>0$, and, for this m update correspondingly the δ^{ij}_{km}'s for $k\in N$. The above process can be repeated until no more variables can be fixed.

Like with the bound $LB^1(S)$, a bound $LB^2(S)$ can be obtained for any vector (u,v) (not necessarily (u^S,v^S)).

Before finishing this section it is important to point out that the role of of the method that we propose for obtaining the new lower bound $L(\widehat{u},\widehat{v})$, goes beyond the possibility of extending the termination criterion of Algorithm 3.4. In fact, the most important feature of this method is that it provides with a faster convergence of the sequence of points generated for solving the Lagrangean dual. In particular, at any iteration t of the subgradient optimization algorithm when the bound $L(\widehat{u},\widehat{v})$ is calculated, instead of using the the subgradient $\gamma(u^t,v^t)$ as the direction to obtain the next point the next point (u^{t+1},v^{t+1}), we set $(u^{t+1},v^{t+1})=(\widehat{u},\widehat{v})$. As we will see in the computational experiments section, in this way we generate a much more stable sequence of points $\{(u^t,v^t)\}_t$, which converges much faster than the sequence generated by a standard sugbradient optimization algorithm.

5. Reduction Tests

Another way of reducing the size of the formulation MP, which is complementary to CG, is to develop reduction tests capable of eliminating variables that either do not appear, or that take value one in an optimal solution. In this section, we develop a simple reduction test which is able to determine hubs that are going to be closed, and hubs that are going to be open in an optimal solution of a given instance. This reduction test uses information obtained from the Lagrangean function for estimating the location and traffic costs, in case that a node $l \in N$ is chosen to become a hub, and excluded from being a hub, respectively. If for some of the two alternatives the estimated cost is greater than the value of the best known solution, then node m will be set to the other alternative. In particular, given a multiplier vector (u,v), a subset $S \subset N$ of candidate hub nodes, and a node $m \in N \setminus S$, we define the estimated cost $\Delta^1_m(u,v,S)$ in the following way,

$$\Delta^1_m(u,v,S) = LC_m(u,v,S) + \left(f_m - \sum_{j\in N} (u_{mjm} + v_{jmm}) - \xi_m(u,v) \right) + L_x(u,v)$$

were,

$$\begin{aligned} LC_m(u,v,S) = \min \quad & \sum_{k\in S} \left(f_k - \sum_{j\in N} (u_{kjk} + v_{jkk}) - \xi_k(u,v) \right) z_{kk} \\ \text{s.t.} \quad & \sum_{k\in S} b_k z_{kk} \geq D - b_m \\ & z_{kk} \in \{0,1\} \qquad \forall\, k \in S \end{aligned}$$

Note that $\Delta^1_m(u,v,S)$ is the value of expression $L(u,v)$ when i) we locate a hub at node m, and ii) we restrict the set of additional candidate hubs to S.

Similarly, for a node $m \in S$, we define the estimated cost $\Delta_m^0(u, v, S)$ as:

$$\Delta_m^0(u, v, S) = LO_m(u, v, S) + L_x(u, v)$$

$$\text{where} \qquad LO_m(u, v, S) = \min \quad \sum_{k \in S \setminus \{m\}} \left(f_k - \sum_{j \in N} (u_{kjk} + v_{jkk}) - \xi_k(u, v) \right) z_{kk}$$

$$\text{s.t.} \quad \sum_{k \in S \setminus \{m\}} b_k z_{kk} \geq D$$

$$z_{kk} \in \{0, 1\} \qquad \forall\, k \in S \setminus \{m\}$$

Now $\Delta_m^0(u, v, S)$ is the value of $L(u, v)$ when in L_z we exclude node m from being a hub.
The following result provides a reduction test for fixing the status of hubs.

Proposition 6 *Let $\bar{\eta}$ be a known upper bound on the optimal solution value η^*, (u, v) a multipliers vector and $S \subseteq N$ containing the optimal set of hubs. Then*

- *If there exists $m \in N \setminus S$ such that $\Delta_m^1(u, v, S) > \bar{\eta}$, then $z_{mm}^* = 0$.*
- *If there exists $m \in S$ such that $\Delta_m^0(u, v, S) > \bar{\eta}$, then $z_{mm}^* = 1$.*

Proof

- Note that $\Delta_m^1(u, v, S)$ is a lower bound on the objective function value when a hub is located at node m. Therefore, if $\Delta_m^1(u, v, S) > \bar{\eta}$, it holds that $z_{mm}^* = 0$ in any optimal solution.
- Similarly, $\Delta_m^0(u, v, S)$ is a lower bound on the objective function value when no hub is located at node m. Therefore, if $\Delta_m^0(u, v, S) > \bar{\eta}$, it holds that $z_{mm}^* = 1$ in any optimal solution.

■

The above result is used in the following way. At the beginning of the subgradient method we consider all possible nodes as candidate hub nodes, that is $S = N$. Then, every time that we obtain an improved lower bound we apply the reduction test for every node $m \in S$ and we eliminate a node m from S if $\Delta_m^1(u, v, S) > \bar{\eta}$, or we fix $z_{mm} = 1$ if $\Delta_m^0(u, v, S) > \bar{\eta}$. Note that by applying the reduction test in this way we ensure that S always contains the optimal set of hubs. When some node m is eliminated from the set of candidate hubs, the set S is updated to $S \setminus \{m\} \subsetneqq N$, and the expression of the Lagrangean function is updated accordingly since we only consider candidate hubs, $k \in S$. Now we denote $L(u, v, S)$ to the updated Lagrangean function. For evaluating $L_z(u, v, S)$ a smaller number of knapsack problems have to be solved. Moreover, since for each element that is removed from S there is an important reduction on the number of x variables, the evaluation of $L_x(u, v, S)$ can be done much more efficiently.

6. Branch and Price

As we will see in the Computational Experiments section for some of the test instances the column generation algorithm enables us to prove the optimality of the best found solution. When optimality cannot be proven, most often the percent duality gap is very small. However, the goal of this work is to propose an exact algorithm that proves the optimality of the obtained solutions. Thus, when the column generation algorithm does not prove optimality of the best solution found we resort to an enumeration algorithm that we describe next. The algorithm consists of three phases. In the first one we perform a partial enumeration in which we enhance the application of the elimination tests so as to reduce the number of variables to branch on, and the sizes of the subproblems in subsequent phases. In the second phase we branch on the hub location variables (z_{kk}'s) that have not been fixed, and in the third phase we branch in the assignment variables (z_{ik}'s).

Let S denote the set of potential hubs at the end of Algorithm 3.4. In the partial enumeration phase, for each $k \in N$ such that the variable z_{kk} has not been fixed so far, we temporary fix $z_{kk} = 0$, $z_{ik} = 0$, $\forall i \in N$ (leaving unfixed the remaining location variables of N), and we solve the resulting Lagrangean dual with set of potential hubs $S \setminus \{k\}$. If the obtained lower bound, lb_k^0, is greater than or equal to the current best upper bound, we set $z_{kk} = 1$. Otherwise, we temporary fix $z_{kk} = 1$ (leaving unfixed the remaining variables of N) and we solve the resulting Lagrangean dual with set of potential hubs $S \cup \{k\}$. Now if the obtained lower bound, lb_k^1, is greater than or equal to the current best upper bound, we set $z_{kk} = 0$, and $z_{ik} = 0$, $\forall i \in N$. In what follows, N denotes the updated set of indices corresponding to the location variables that have not been fixed so far.

At any node of the second phase, we run again Algorithm 3.4 with the current set of variables. That is, we solve the pricing problem for generating new columns, until the node can be eliminated (the associated lower bound is greater than or equal to the incumbent value), or until no more new columns can be generated. In the second phase we explore the search tree in a depth search first fashion. Hence, when no more columns can be generated at a given node, again we branch on one of the remaining unfixed location variables. If no unfixed location variable exists we backtrack to the next unexplored node (even if the node cannot be eliminated). When we backtrack to an unexplored node, we keep all generated columns excepting those incompatible with the new node. The order in which variables are selected for branching in the second phase is determined by the result of the first phase. For each $m \in N$, we consider $\Delta_m = \max\{lb_m^0, lb_m^1\}$, and we branch on variable $k \in N$ such that $\Delta_k = \max\{\Delta_m : m \in N\}$. Then, if $\Delta_k = lb_k^0$, we explore the branch corresponding to $z_{kk} = 1$; otherwise, we explore the branch corresponding to $z_{kk} = 0$.

When all the nodes of the second phase have been explored, but some of them have not been eliminated, we start the third and final phase. In this phase we branch on the assignment variables, since for each un-eliminated node, all the location variables are already fixed. It is worth mentioning that even if all the location variables are fixed the resulting subproblem is a quadratic semiassignment problem with capacities, which is not an easy problem. The order in which non-hub unassigned nodes are considered for branching is by decreasing values of their total outgoing flow O_i. That is we branch first on the variables that will consume more capacity of the hubs they are assigned to. The set of potential hubs to which the selected branching variable can be assigned, H_i, is given by the hubs that are open in the path of the search tree from the root node to the current node, whose current available capacity is greater than or equal to O_i. The part of the tree corresponding to the third phase is not binary. When we select a variable for branching we generate as many branches as feasible potential hubs there exist for the assignment of the variable (at most $|H_i|$). The order in which the branches are explored is by increasing values of the assignment costs c_{ik}, $k \in H_i$.

7. Computational Experience

In this section we describe the computational experiments that we have run for evaluating the performance of the proposed methods and we present and analyze the obtained numerical results.

All the algorithms have been coded in C and run on a HP mobile workstation with a processor Intel(R) Core(TM)2 T7600 with 2.33GHz and 3.37 GB of RAM memory. The knapsack problems have been solved with the Martello et al. [16] code.

We have used the 56 benchmark instances of [8]. They are divided in the following three sets: the first one contains the twenty small-size instances of sizes n =10, 20, 25, 40, and 50 used in [11]. The second set contains the medium-size instances of sizes $n = 60, 70, 75, 90$ generated in [8]. Finally, the third set contains the eight large-size instances of 100 and 200 nodes used in [11], as well as four instances of each size $n = 125, 150, 175$ generated in [8]. There are four instances for each of the considered sizes. For a fixed size, each of the four instances corresponds to a different combination of characteristics for the fixed set-up costs and the capacities for the hubs as described in [11]. In particular, there are two possibilities for the set-up costs: tight (T), when they increase with the amount of flow generated in the node, and loose (L), when the instances do not exhibit this characteristic. There are also two possibilities for the capacities of the hubs: tight (T) and loose (L), depending on how tightly constrained the instances are. We denote each instance as nFC where n is the problem size, F is the option for the set-up costs and C the type of capacity.

In all the experiments, the subgradient optimization 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 below a threshold value, i.e. $\left|z^* - z_D^k\right| < \epsilon$; *iii*) The improvement on the lower bound after t consecutive iterations is below a threshold value γ; *iv*) The maximum number of iterations $Iter_{max}$ is reached.

After some tuning, we set the following parameter values: $Iter_{max} = 1000$, $t = 200$, $\epsilon = 10^{-3}$, and $\gamma = 0.015\%$. Every 5 iterations of the subgradient optimization algorithm we run the simple heuristic of [8] trying to find a feasible solution to improve the current upper bound. The heuristic is also used each time that we obtain an improved lower bound. In the partial enumeration phase, for a faster convergence (and given that the exact enumeration follows after) we have changed the parameters of the subgradient optimization. In particular, the parameters that we have used during the partial enumeration phase are $Iter_{max} = 500$, $t = 100$, $\epsilon = 10^{-3}$, and $\gamma = 0.015\%$.

Our preliminary computational tests focused on the comparison between the two lower bounds $LB^1(S)$ and $LB^2(S)$. This test also allowed us to see how the stabilization method influenced the speed of convergence of the subgradient optimization when the lower bound $LB^2(S)$ was used. Figure 1 compares the values of the Lagrangean Relaxation of RPM (which are not necessarily valid lower bounds) with: *i*) the lower bounds $LB^1(S)$; *ii*) the lower bounds $LB^2(S)$, without using the stabilization method; and, *iii*) the lower bounds $LB^2(S)$, using the stabilization method. The bounds $LB^1(S)$ and $LB^2(S)$ are obtained every 80 iterations of subgradient optimization applied to the Lagrangean Relaxation of the RMP. For illustrative purposes we have arbitrarily selected one medium-size instance, in particular instance 70TT, and the first global iteration of the CG algorithm, although the results are qualitatively very similar for all the considered instances.

As was expected $LB^2(S)$ outperformed considerably $LB^1(S)$ in the vast majority of the cases. Also, the application of the stabilization method improved notably the convergence of the algorithm and also had a beneficial effect on the quality of the bounds that were obtained, which were very close to the values (not necessarily valid lower bounds) of the Lagrangean relaxation of the RPM.

Given the superiority of $LB^2(S)$ over $LB^1(S)$ we rejected $LB^1(S)$ from any further consideration, and only lower bound $LB^2(S)$ was used in subsequent experiments.

We have implemented five different versions of our CG algorithm, all of which are variations of Algorithm 3.4. The versions differ from each other in how the extended termination criterion, the reduction tests and the stabilization method are combined for enhancing the performance of the basic algorithm. This will allow us to see which is the contribution of each of these components to the overall efficiency of the algorithm. The versions are the following:

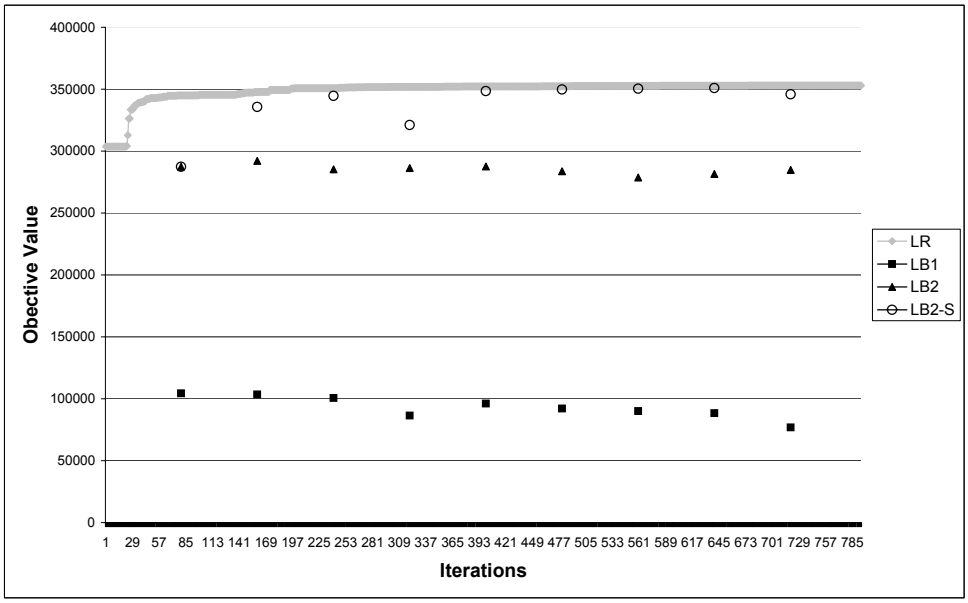

Figure 1: Comparison between $LB^1(S)$ and $LB^2(S)$

- CG^0: Basic version of column generation algorithm 3.4 that does not use any lower bound. The termination criterion is when the pricing problem does not generate additional columns. Since it does not use any lower bound, the tests for eliminating variables are not used.

- CG^1: Column generation algorithm that uses the lower bound $LB^2(S)$. The bound is calculated every 80 iterations of the subgradient optimization algorithm with the current dual variables. The lower bound is only used to extend the termination criterion of the column generation algorithm so as to allow termination when $|z_D^S - LB^2(S)| < \epsilon$, but no elimination tests are applied.

- CG^2: Like GC^1 but we also apply the reduction tests for fixing variables. The reduction tests are applied each time the lower bound is evaluated.

- CG^3: Like CG^1 but using the stabilization method. Every time the lower bound $LB^2(S)$ is obtained, instead of using the subgradient for updating the current dual variables, we update the dual variables to $(\widehat{u}, \widehat{v})$ as described in Section 4. In this version we do not apply the reduction tests.

- CG^4: Like CG^3, applying also the reduction tests every time the lower is evaluated.

In all the versions, the initial set of columns is generated with the GRASP heuristic of the Appendix. Since this heuristic is not deterministic, the results of the different versions may vary in

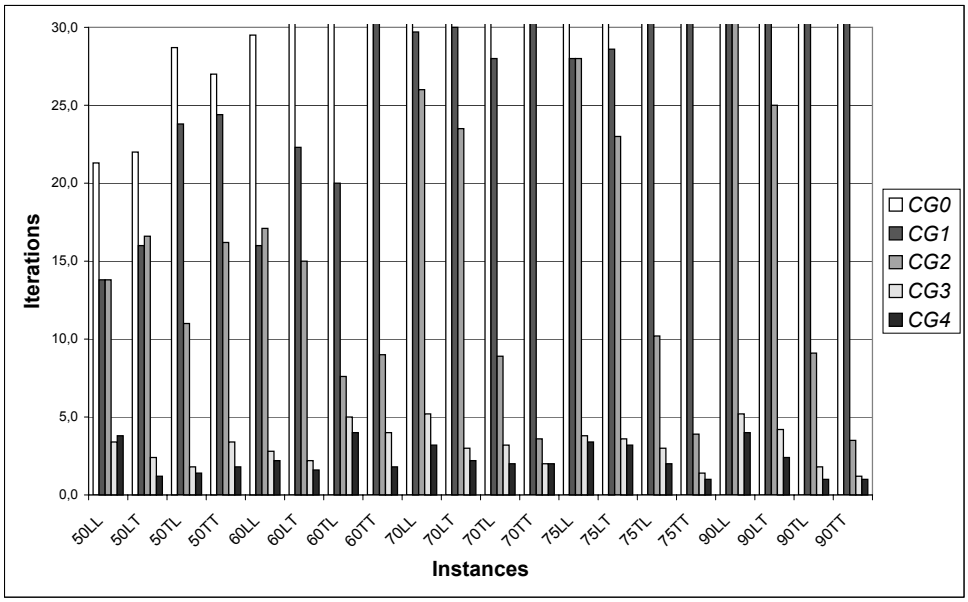

Figure 2: Comparison between the number of global iterations for the different versions of GC.

different runs. For this reason, all our results correspond to averages over 5 runs of each instance. Figures 2 and 3 allow to compare the different versions between them. These two figures give, for the five versions, the number of global iterations and the required cpu time (in seconds), respectively, over the set of medium size instances. In Figure 2 all values over 30 have been truncated, so that the scale can be better appreciated. Similarly, in Figure 3 all values over 600 have been truncated. While CG^1 improves considerably the basic version GC^0 both in terms of global number of iterations and required cpu times, extending the termination criterion is clearly not enough so as to obtain an efficient algorithm since CG^1 exceeds the limit of 40 iterations for most instances over 60 nodes and it is over 4000 seconds of cpu time, for instances 90TL and 90TT. In CG^2 the effect of the reduction tests over the extended termination criterion is remarkable for instances TL and TT but, even if it generally yields improvements for instances LL and LT, its impact is not so important for these instances. On the contrary, the impact of the stabilization method is crucial on all types of instances for the efficiency of the algorithm. As can be seen, the versions that use the stabilization method (CG^3 and CG^4) are considerably more efficient that the ones that do not use this method, both in terms of the number of global iterations and of total cpu time. Moreover, while the number of global iterations of the versions that do not use the stabilization method seem to increase nearly exponentially with the sizes of the instances, the number of global iterations of the versions that use the stabilization method do not seem to depend on the size of the instance, but on the type of instance. In particular, CG^4 always required less than 5 global iterations, whereas CG^3 is slightly

over 5 global iterations only for instances 70LT and 90LT. This supremacy of the versions that use the stabilization method can also be appreciated in terms of the required cpu times, since CG^4 never exceeds 100 seconds of cpu time, and CG^3 is above 100 seconds of cpu time only for instances 90LT and 90TT. Finally, we can observe that the inclusion of the reduction tests on the algorithm is beneficial since it produces further improvements on the algorithm in terms of efficiency. In particular, excepting for two instances (instances 50LL and 60LL) where CG^4 is slightly less time consuming than CG^3, the latter outperforms the former, and the superiority increases as the sizes of the instances increase. Therefore, for the comparison with the Lagrangean Relaxation of [8] and for the implementation of the Branch-and-Price algorithm we have selected version GC^4.

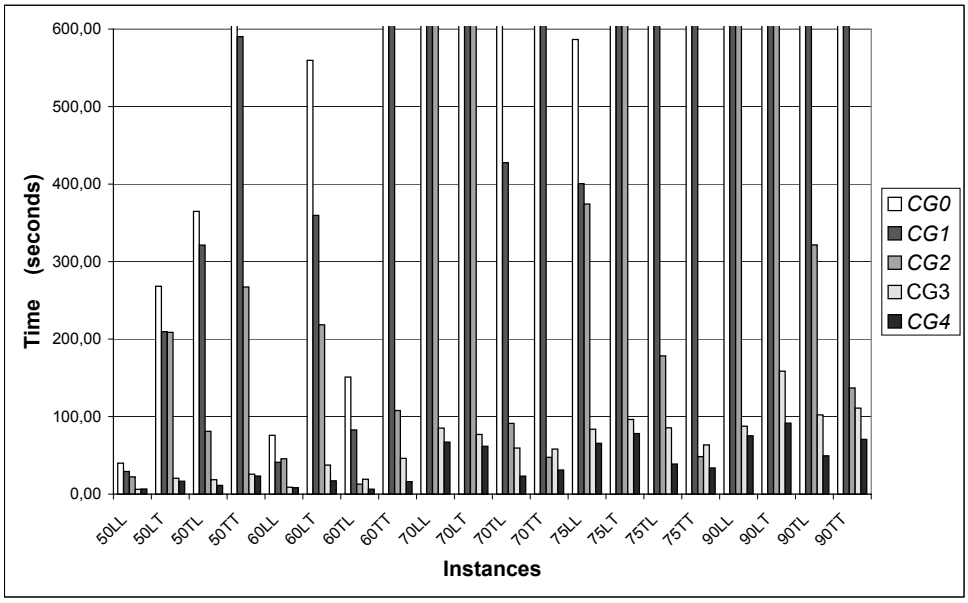

Figure 3: Comparison between the cpu times for the different versions of GC.

Table 1 gives the detailed results of CG^4 and compares them with the results of the Lagrangean relaxation of [8] (denoted LR) for all the considered instances. The first column, gives the name of the instance. The next two columns, under *Avg. %Dev.*, depict the average percent deviation of the optimal solution (obtained with the exact branch–and–price algorithm) relative to the bound obtained with the Lagrangean relaxation from [8] for LR and CG^4. The next two columns under the heading *Avg. Time* give the cpu time in seconds until termination both for LR and CG^4. The next two columns, *Avg. Iter.* and *Avg. % Cols.*, give for CG^4 the number of global iterations needed to terminate and the number of columns generated (expressed as a percentage of the total number of possible columns $|N|^4$), respectively. Finally, the last two columns under *Fixed Hubs* depict the average number of hubs that were fixed with the reduction tests in LR and CG^4, respectively. The

results obtained with CG^4 are remarkable and indicate that it clearly outperforms LR. In terms of the obtained lower bounds, both methods yield more or less similar bounds. Observe that for CG^4 46 of the 56 considered instances the average percent duality gap is below 1%. The largest percent gap at termination is 3.18, and there are only nine instances for which this percent gap is over one percent. These results improve slightly the results of LR, with a largest percent gap of 3.38 and 11 instances with a percent gap over 1%. This similarity in the lower bounds was expected since, as it has already been commented, in the end both methods solve the lagrangean dual of MP. Thus, the differences are related either to small changes due to the convergence criteria, or, most often, to the upper bound that is obtained at the inner iterations of subgradient optimization, which allows to apply more efficiently the reduction tests. However, the most clear indicator of the superiority of CG^4 over LR is the required cpu time. As can be seen, excepting for a few very small instances and for instance 125TL (where RL takes 89.11 secs. and CG^4 takes 103.15 secs.), the times required by CG^4 are considerably smaller that the ones of LR, and this difference becomes more evident as the sizes of the instances increase. The instance that takes more time, both for LR and CG^4 is, 200LT which requires 23212.58 seconds with LR and 3969.97 (just over 17%) with CG^4. This efficiency of CG^4 is indeed due to the small number of global iterations required by CG^4 which, as we have seen when comparing the different versions of the column generation algorithm, is in its turn, due to the stabilization method that we apply. Note that the number of columns that are generated throughout the column generation is only a very small fraction of the total number of potential columns that exist in MP. Observe also, how this fraction becomes smaller as the sizes of the instances increase, and is below 0.20% for the larger instances. This means that the sizes of the RPM's that we solve tend to become moderate during the column generation process even for the larger instances. Finally we can see that, the reduction tests have more impact on LR than on CG^4 for the smaller instances, while this tendency is inverted as the size of the instances increase. Nevertheless, both for LR and CG^4 we can appreciate that a considerable number of hubs are fixed for medium-size and large instances.

The results of the exact branch–and–price algorithm are given in Table 2. As before, the first column, gives the name of the instance. The next three columns under the heading *Root node* give information of the root node. In particular, column *Fixed* gives the number of hubs that were fixed (either opened or discarded) with the column generation algorithm, column *Dev* gives the percent deviation from the optimal solution at termination of CG^2, and and *time* the required cpu time. The information under *Dev* and *time* was already depicted in Table 1, but it is repeated here for a better visualization of the results. Columns under *Partial Exp.* give information on the partial

	Avg. %Dev.		*Avg. Time*		*Avg. Iter.*	*Avg. %Cols.*	*Fixed Hubs*	
Prob	*LR*	CG^4	*LR*	CG^4	CG^4	CG^4	*LR*	CG^4
10LL	0.00	0.00	0.03	0.07	3.0	6.58	2	0
10LT	0.00	0.01	0.03	0.05	2.0	6.93	4	1
10TL	0.00	0.00	0.06	0.06	2.0	7.67	5	0
10TT	0.00	0.01	0.05	0.05	2.0	5.99	7	1
20LL	0.00	0.00	0.38	0.23	3.8	2.84	16	0
20LT	0.83	0.83	1.64	0.93	1.6	2.26	6	7
20TL	0.00	0.00	0.19	0.21	3.0	3.04	17	0
20TT	0.00	0.00	0.16	0.38	1.0	2.25	17	15
25LL	0.00	0.02	1.97	0.64	4.0	1.50	8	0
25LT	0.04	0.01	2.55	1.19	1.8	1.70	10	14
25TL	0.00	0.00	1.00	0.64	1.6	1.29	12	20
25TT	0.34	0.25	2.47	1.57	1.8	1.43	11	13
40LL	0.07	0.08	7.24	5.21	2.0	0.79	21	30
40LT	0.08	0.06	6.81	6.92	2.6	0.60	25	27
40TL	0.00	0.01	1.59	1.93	1.0	0.40	37	36
40TT	0.50	0.59	11.39	4.98	1.0	0.50	32	32
50LL	0.00	0.02	13.00	6.60	3.8	0.56	43	1
50LT	0.43	0.57	63.98	16.65	1.2	0.51	7	31
50TL	0.80	0.73	20.02	11.30	1.4	0.31	41	39
50TT	2.93	2.98	77.16	23.37	1.8	0.57	14	27
60LL	0.00	0.00	21.56	8.35	2.2	0.38	40	33
60LT	0.45	0.30	94.80	17.15	1.6	0.33	14	49
60TL	0.00	0.00	2.45	6.39	4.0	0.19	58	47
60TT	0.81	0.73	33.39	16.10	1.8	0.21	54	50
70LL	0.82	0.79	145.05	67.12	3.2	0.21	19	31
70LT	0.70	0.70	185.70	61.73	2.2	0.28	33	49
70TL	0.10	0.06	15.50	23.21	2.0	0.16	63	64
70TT	0.33	0.34	48.99	31.02	2.0	0.26	56	61
75LL	0.05	0.00	145.44	65.54	4.0	0.19	29	73
75LT	0.28	0.34	203.43	78.14	3.2	0.25	26	53
75TL	0.12	0.23	44.83	38.76	2.0	0.13	69	67
75TT	2.35	2.33	85.06	33.78	1.0	0.16	54	64
90LL	0.00	0.03	237.95	75.28	4.0	0.16	50	26
90LT	0.21	0.23	515.84	91.56	2.4	0.17	26	67
90TL	0.93	0.84	110.39	49.46	1.0	0.09	74	78
90TT	0.29	0.50	141.34	70.64	1.0	0.15	75	82
100LL	0.63	0.70	873.80	220.68	5.0	0.17	16	49
100LT	1.11	0.30	1039.60	139.01	2.0	0.15	5	78
100TL	3.38	3.18	397.02	77.53	2.5	0.07	61	84
100TT	0.85	0.91	347.87	77.10	1.0	0.10	86	89
125LL	1.19	1.03	2731.23	618.74	4.4	0.10	24	65
125LT	0.42	0.30	2096.22	337.64	3.2	0.12	85	98
125TL	0.00	0.00	89.11	103.15	2.0	0.07	123	123
125TT	1.28	0.30	287.50	118.67	1.6	0.03	119	120
150LL	0.51	0.33	6671.82	1329.39	4.5	0.08	22	99
150LT	2.32	1.80	5852.48	776.19	1.3	0.10	6	81
150TL	1.43	1.42	1009.87	313.61	1.1	0.04	118	138
150TT	2.02	1.68	1802.13	348.79	1.0	0.05	103	133
175LL	1.01	0.36	10352.05	1573.83	5.7	0.05	29	84
175LT	1.53	1.28	13172.84	993.38	1.2	0.05	21	111
175TL	0.14	0.21	1019.50	474.70	2.0	0.04	144	164
175TT	0.62	0.85	1856.37	472.88	2.1	0.04	140	158
200LL	0.64	0.37	19590.75	3747.26	6.5	0.06	28	111
200LT	2.83	1.97	23212.58	3969.97	5.0	0.09	11	95
200TL	0.04	0.23	2801.48	1682.64	7.2	0.03	189	175
200TT	0.88	0.55	3111.90	810.81	2.4	0.04	154	179

Table 1: Column Generation results

enumeration phase of the exact algorithm. Column under *Fixed* gives the number of hubs that were fixed (either opened or discarded) at the end of this phase (the number of hubs fixed during this phase can be obtained by subtracting the figure in column *Fixed* under *Root Node* from the number in the entry); and column under *time* gives the cpu time spent by the partial enumeration phase. The next three columns under *Branch Location* give information on the second phase of the search tree, i.e., when the search tree enumerates the location variables. In particular, column under *Nodes* gives the number of nodes generated in this phase, column *Dev* gives the average percent deviation between the optimal solution and the current lower bound at termination of the second phase, and column under *time* gives the time required for the branch–and–bound algorithm in this phase. The next three columns, under *Branch Assignment* give a similar information, relative to the second phase where the enumeration on the assignment variables takes place. Again, *Nodes* gives the number of nodes generated in this phase, column *Dev* gives the average percent deviation between the optimal solution and the current lower bound at termination of the third phase, and column under *time* gives the time required for the branch–and–bound algorithm in this phase. Finally, column *Optimal* gives the optimal value of the instance and column *Total time* gives the total time until optimality of the solution was proven.

As can be seen in Table 2, 14 instances could be optimally solved without resorting to the complete enumeration phase, and four more instances only explored nodes associated with the location variables. For the instances where the complete enumeration took place, in general, the number of explored location nodes was smaller than the number of explored allocation nodes. Indeed, this could be expected, since the number of the later variables exceeds notably the number of location variables in the considered formulation. As for the required cpu times, observe that 39 out of the 56 considered instances were optimally solved in less than 600 seconds of cpu time; seven more instances were solved in less than one hour and three more instances could be solved within a time limit of two hours of cpu time. As can be seen only eight instances required a total time over 7200 seconds of cpu time. There is one single instance in the considered set, instance 200LL, for which optimality of the current solution could not be proven, since we run out of memory. At that point the percent gap between the current upper and lower bound was 0.2%, so that it is most likely that the best-known solution is in fact optimal for the instance.

Summarizing, the results of Table 2 indicate that we have been able to solve exactly 55 out of the 56 considered instances. The percent gap of the only instance for which optimality of the best found solution could not be proven is 0.2%. It is worth mentioning that the sizes of these instances go up to 200 nodes. These seem to be the largest instances that have been solved exactly for the

CHLPSA. In our opinion, the obtained results are very good taking into account the difficulty of the problem and the sizes of the considered instances.

8. Conclusions

In this paper we have presented a branch–and–price algorithm for the Capacitated Hub Location Problem with Single Assignment. We have used Lagrangean relaxation to obtain tight lower bounds of the Restricted Master Problem (RPM), and we have shown how to solve the pricing problem for finding new columns to be added to the RMP. The process that we use to obtain a tight lower bound has allowed us to apply a stabilization method that improves considerable the overall efficiency of the solution algorithm. The numerical results on a battery of benchmark instances of up to 200 nodes assess the efficiency of the proposal. To the best of our knowledge, the considered instances are the largest instances that have been solved exactly for the CHLPSA.

Acknowledgements

This research has been partially supported by grant MTM2006-14961-C05-01 of the Spanish Ministry of Education and Science. The research of the first author has been partially supported through grant 197243/217997 from CONACYT, México. These supports are gratefully acknowledged.

References

[1] S. Alumur and B.Y. Kara. Network hub location problems: The state of the art. *European Journal of Operational Research*, 190(1):1–21, 2008.

[2] T. Aykin. Lagrangian relaxation based approaches to capacitated hub-and-spoke network design problem. *European Journal of Operational Research*, 79:501–23, 1994.

[3] T. Aykin. Networking policies for hub-and-spoke systems with applications to the air transportation system. *Transportation Science*, 3:201–221, 1995.

[4] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column geneartion for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998b.

[5] N.L. Boland, A.T. Ernst, M. Krishnamoorthy, and J. Ebery. Preprocessing and cutting methods for multiple allocation hub location problems. *European Journal of Operational Research*, 155(3):638–653, 2004.

	Root Node			*Partial Exp.*		*Branch Location*			*Branch Assignment*			*Optimal*	*Total*
Prob	*Fixed*	*Dev*	*time*	*Fixed*	*time*	*Nodes*	*Dev*	*time*	*Nodes*	*Dev*	*time*	*Solution*	*time*
10LL	0	0.00	0.09	0	0.00	0	0.00	0.00	0	0.00	0.00	224250.05	0.09
10LT	1	0.01	0.14	10	0.11	0	0.00	0.00	0	0.00	0.00	250992.26	0.27
10TL	0	0.00	0.16	0	0.00	0	0.00	0.00	0	0.00	0.00	263399.94	0.16
10TT	1	0.01	0.09	10	0.05	0	0.00	0.00	0	0.00	0.00	263399.94	0.14
20LL	0	0.00	0.28	0	0.00	0	0.00	0.00	0	0.00	0.00	234690.96	0.28
20LT	7	0.85	1.02	17	1.74	3	0.00	0.22	0	0.00	0.00	253517.40	2.97
20TL	0	0.01	0.27	20	0.52	0	0.00	0.00	0	0.00	0.00	271128.18	0.78
20TT	14	0.00	0.42	0	0.00	0	0.00	0.00	0	0.00	0.00	296035.40	0.42
25LL	0	0.00	0.77	25	2.59	0	0.00	0.00	0	0.00	0.00	238977.95	3.38
25LT	15	0.04	1.11	25	1.13	0	0.00	0.00	0	0.00	0.00	276372.50	2.25
25TL	12	0.00	0.58	25	0.25	0	0.00	0.00	0	0.00	0.00	310317.64	0.83
25TT	13	0.25	1.41	20	3.49	3	0.00	0.36	0	0.00	0.00	348369.15	5.25
40LL	29	0.08	3.78	40	4.22	0	0.00	0.00	0	0.00	0.00	241955.71	8.17
40LT	13	0.04	6.97	40	3.80	0	0.00	0.00	0	0.00	0.00	272218.32	11.53
40TL	34	0.00	1.59	0	0.00	0	0.00	0.00	0	0.00	0.00	298919.01	1.59
40TT	25	0.68	4.09	40	2.67	0	0.04	0.00	19	0.00	2.13	354874.10	9.69
50LL	0	0.00	5.00	50	12.24	0	0.00	0.00	0	0.00	0.00	238520.59	17.36
50LT	32	0.49	11.42	49	17.31	3	0.00	2.08	0	0.00	0.00	272897.49	30.81
50TL	43	0.80	5.66	49	3.91	3	0.00	0.41	0	0.00	0.00	319015.77	9.97
50TT	25	2.72	18.97	43	88.56	37	0.68	140.44	131	0.00	78.99	417440.99	326.97
60LL	24	0.04	6.86	60	16.19	0	0.00	0.00	3	0.00	0.09	225917.21	26.55
60LT	46	0.28	17.45	60	9.44	0	0.00	0.00	5	0.00	0.08	253616.51	30.22
60TL	47	0.00	5.27	60	1.55	0	0.00	0.00	3	0.00	0.02	252496.66	7.55
60TT	52	0.83	11.22	59	13.58	3	0.11	1.56	55	0.00	4.05	351203.17	30.41
70LL	38	0.80	50.05	67	161.20	7	0.80	25.47	103	0.00	2.14	236817.35	238.86
70LT	50	0.84	38.79	65	145.13	11	0.36	32.27	349	0.00	312.14	256920.45	528.34
70TL	65	0.00	19.57	70	3.88	0	0.00	0.00	3	0.00	0.08	271283.82	24.56
70TT	61	0.29	26.53	70	4.50	0	0.17	0.00	89	0.00	94.37	387380.20	130.58
75LL	55	0.02	51.56	75	66.37	0	0.00	0.00	3	0.00	0.03	238024.22	118.05
75LT	55	0.33	58.47	74	43.03	3	0.05	7.60	7	0.00	7.82	256188.12	116.92
75TL	69	0.27	30.27	75	6.96	0	0.01	0.00	3	0.00	0.11	303363.55	38.82
75TT	64	2.50	22.76	73	55.75	5	0.02	4.38	9	0.00	10.15	347189.81	93.04
90LL	6	0.01	70.79	90	611.39	0	0.00	0.00	3	0.00	0.08	224195.72	683.84
90LT	71	0.18	100.67	90	53.53	0	0.03	0.00	73	0.00	53.01	246026.24	220.14
90TL	78	0.85	41.19	90	37.63	0	0.25	0.00	607	0.00	419.07	281561.56	507.75
90TT	82	0.52	52.84	90	12.96	0	0.33	0.00	1585	0.00	427.33	337008.93	498.81
100LL	48	0.69	311.59	97	1390.45	7	0.02	220.04	4	0.00	0.27	246713.97	1922.35
100LT	77	0.30	134.93	97	314.39	7	0.30	33.82	175	0.00	394.96	256155.33	878.09
100TL	84	3.43	60.77	97	114.04	9	0.48	59.58	73	0.00	43.88	362950.09	278.27
100TT	89	0.79	75.19	98	93.57	5	0.50	24.38	381	0.00	630.44	474068.96	823.58
125LL	59	1.07	687.00	118	4207.13	17	0.02	1558.96	61	0.00	11.31	239889.33	6464.41
125LT	101	0.46	172.56	124	204.41	3	0.03	22.33	141	0.00	331.86	251259.16	731.16
125TL	122	0.01	72.14	125	1.39	0	0.00	0.00	3	0.00	0.97	246486.69	84.13
125TT	120	0.29	75.06	125	6.89	0	0.26	0.00	229	0.00	431.95	291807.35	517.27
150LL	90	0.39	1268.48	145	5125.25	11	0.00	858.07	3	0.00	0.95	234765.44	7252.71
150LT	97	1.63	785.98	140	6179.12	39	1.63	4844.90	74	0.00	679.11	249797.49	12489.11
150TL	136	0.99	190.25	148	260.42	5	0.21	59.53	365	0.00	1637.00	262543.08	2147.20
150TT	130	1.73	314.27	144	936.59	15	0.30	401.90	1646	0.00	7326.91	322976.47	8979.67
175LL	110	0.32	1294.41	174	5349.36	3	0.01	395.29	213	0.00	133.35	227997.58	7172.41
175LT	125	1.23	1356.60	167	6361.26	23	0.24	3586.79	121	0.00	136.76	251540.80	11441.40
175TL	167	0.30	422.90	175	170.79	0	0.01	0.00	3	0.00	0.97	244860.41	613.51
175TT	151	1.34	672.45	171	1532.66	9	0.43	309.60	1521	0.00	6177.59	312193.78	8692.29
200LL	111	0.37	3747.26	130	6434.74	104	0.20	–.–	—	-.–	–.–	——.–	—-.–
200LT	104	1.36	3888.97	178	44731.13	45	0.13	27605.94	1553	0.00	5772.66	267218.35	81998.70
200TL	178	0.34	1348.89	198	1567.23	5	0.00	136.38	9	0.00	2.00	273443.81	3054.50
200TT	181	0.46	738.24	199	735.83	3	0.21	102.68	757	0.00	5551.38	290582.04	7128.13

Table 2: Branch and Price results

[6] J. F. Campbell. Integer programming formulations of discrete hub location problems. *European Journal of Operational Research*, 72:387–405, 1994.

[7] J.F. Campbell, A. Ernst, and M. Krishnamoorthy. Hub location problems. In H. Hamacher and Z. Drezner, editors, *Location Theory: Applications and Theory*, pages 373–406. Springer-Verlag, 1987.

[8] I. Contreras, J. Díaz, and E. Fernández. Lagrangean relaxation for the capacitated hub location problem with single assigment. *to appear in OR Spectrum*.

[9] M.G. Costa, M.E. Captivo, and J. Climaco. Capacitated sinlge allocation hub location problem - a bi-criteria approach. *Computers and Operations Research*, 35(11):3671–3695, 2008.

[10] J. Ebery, M. Krishnamoorthy, A. T. Ernst, and N. Boland. The capacitated multiple allocation hub location problem: Formulations and algorithms. *European Journal of Operational Research*, 120(3):614–631, 2000.

[11] A.T. Ernst and M. Krishnamoorthy. Solution algorithms for the capacitated single allocation hub location problem. *Annals of Operational Research*, 86:141–159, 1999.

[12] H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quaterly*, 2:83–97, 1955.

[13] M. Labbé, H. Yaman, and E. Gourdin. A branch and cut algorithm for hub location problems with single assignment. *Mathematical Programming*, 102(2):371–405, 2005.

[14] M. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2005.

[15] A. Marin. Formulating and solving splittable capacitated multiple allocation hub location problems. *Computers and Operations Research*, 32:3093–3109, 2005.

[16] S. Martello, D. Pisinger D, and P. Toth. New trends in exact algorithms for the 0-1 knapsack problem. *European Journal of Operational Research*, 123:325–332, 2000.

[17] I. Rodríguez-Martín and J. J. Salazar-González. Solving a capacitated hub location problem. *European Journal of Operational Research*, 184:468–479, 2008.

[18] M. Sasaki and M. Fukushima. On the hub-and-spoke model with arc capacities constraints. *Journal of the Operations Research Society of Japan*, 46(4):409–428, 2003.

A. GRASP heuristic for obtaining the initial set of columns.

In what follows, solutions are represented by pairs of the form $h = (H, a)$ where $H \subseteq N$ denotes the set of selected sites to locate the hubs and $a : N \to H$ is the assigning mapping, i.e. $a(i) = k$ if customer $i \in N$ is assigned to hub $k \in H$. For any feasible assignment, h_k denotes the available capacity of hub k (i.e. $h_k = b_k - \sum_{i:a(i)=k} d_i$).

The GRASP heuristic consists of a construction phase and an improvement phase. The construction phase is as follows:

Construction phase

At each step of the Construction Phase the unassigned node with the biggest outgoing flow, that is $\hat{i} = max\{O_i : a(i) = null\}$, is assigned to a candidate hub node and several other nodes are also assigned to this candidate hub node. Let $s = (M, a)$ be a partial solution where some clients are not yet allocated (i.e. $a(j) = null$). Nodes are ordered by increasing values of O_i and the assignments are always feasible with respect to the capacity constraints. For each candidate hub node $k \in N \setminus M$ we compute the greedy function:

$$\delta(k) = \frac{f_k + \sum_{j \in Nodes_k} (O_j + D_j) c_{jk}}{|Nodes_k|}$$

where $Nodes_k$ is the list of all unassigned nodes that fit into the hub located at k, assuming that all the nodes in set $Nodes_k$ are assigned to hub k according to the ordering described above. For all $k \in N \setminus M$, $Nodes_k$ always contains $\hat{i}$ and if $\hat{i} \neq k$ it also contains k. In order to achieve a tradeoff between quality and diversity, a partially randomized greedy procedure is considered. The choice of the next node to become a hub node is determined by randomly selecting one element from a Restricted Candidate List (RCL). The RCL is updated at each iteration of the construction phase and contains the best candidates with respect to a threshold value. This threshold value is computed in the following way. Let $\delta(k)_{min} = min\{\delta(k) : k \in N \setminus M\}$ and $\delta(k)_{max} = max\{\delta(k) : k \in N \setminus M\}$. Then, $RCL = \{k : \delta(k)_{min} + (\delta(k)_{max} - \delta(k)_{min})\}$. The construction phase terminates when all nodes are assigned to an open hub node.

Local Search Phase

The Local Search Phase procedure explores two types of neighborhood structures. The neighborhood structures of the first type are related to the assignment of nodes within the set of selected

hubs, whereas the neighborhoods of the second type are related to solutions where the set of open hubs changes. In all cases, we consider solutions within the feasible domain.

The *shift* neighborhood considers all solutions that can be reached from the current one by changing the assignment of exactly one node, whereas the *swap* neighborhood contains all solution that differ from the current one in the assignment of two nodes. Let $s = (S, a)$ be the current solution, then

$$N_{shift}(s) = \left\{ s' = (M, a') : \exists! i \in N, a'(i) \neq a(i) \right\}$$

and

$$N_{swap}(s) = \left\{ s' = (M, a') : \exists i_1, i_2, a'(i_1) = a(i_2), a'(i_2) = a(i_1), a'(i) = a(i), \forall i \neq i_1, i_2 \right\}$$

For exploring N_{shift} we consider all pairs of the form (i, j) where $a(j) \neq i$ and $h_i \geq d_j$. Also, for exploring N_{swap} we consider all pairs of the form (i_1, i_2) where $a(i_1) \neq a(i_2), h_{a(i_1)} + d_{i_1} \geq d_{i_2}$ and $h_{a(i_2)} + d_{i_2} \geq d_{i_1}$. In both cases we perform the first improving move.

We explore two additional neighborhood structures of the second type. They affect the current set of open hubs. The first one considers a subset of feasible solutions that are obtained from the current one by opening a new hub and by assigning some clients to it. Then,

$$N_{open}(s) \subset \left\{ s' = (M', a') : M' = M \cup \{k\}\,; \forall j, a'(j) = r \in M' s.t. \sum_{j:a'(j)=r} O_j \leq b_r, \forall r \in M' \right\}$$

To explore $N_{open}(s)$ all nodes $k \in N \backslash M$ are considered. Again, let $s = (M, a)$ denote the current solution, $a'(k)$ the new assignment and $\hat{h_r}$ the available capacity of hub r. Initially, $a'(p) = a(p)$ for all $p \in N$ and $\hat{h_r} = h_r$ for all $r \in M$. For each potential hub node (i.e. $k \in N \setminus M$), we consider nodes by decreasing order of their O_i values. Node j is reassigned to hub k if $c_{jk} \leq c_{a(j),j}$ and $\hat{h_k} \geq O_j$. If node j is reassigned to hub k we update its assignment and the available capacity of hubs k and $a(j)$.

The second neighborhood structure of the second type, considers a subset of feasible solutions that are obtained from the current one by closing a hub and reassigning its assigned nodes to other open hubs. Then,

$$N_{close}(s) \subset \left\{ s' = (M', a') : M' = M \setminus k; \forall j, a'(j) = r \in M' s.t. \sum_{j:a'(j)=r} O_j \leq b_r, \forall r \in M' \right\}$$

To explore $N_{close}(s)$ all hub nodes $k \in M$ are considered. Again, let $s = (M, a)$ denote the current solution, $a'(k)$ the new assignment and $\hat{h_r}$ the available capacity of hub r. Initially, $a'(p) = a(p)$ for all $p \in N$ and $\hat{h_r} = h_r$ for all $r \in M$. For each hub node (i.e. $k \in M$), we consider its assigned nodes by decreasing order of their O_i values. Node j is reassigned to hub $\hat{m}$, where $\hat{m} = min\left\{c_{jm} : \hat{h_m} - O_j \geq 0, m \in M \setminus k\right\}$. If node j is reassigned to hub $\hat{m}$ we update its assignment and the available capacity of hubs $\hat{m}$ and $a(j)$.

Finally, we obtain the initial set S by considering $S_{ij} = \bigcup_{i=1}^{k} H_k$ for every pair of nodes $i, j \in N$, where $h_1 = (H_1, a_1)$, ..., $h_k = (H_k, a_k)$ denote the k best solutions found by the GRASP heuristic associated with different sets of open hubs.