The Dynamic Uncapacitated Hub Location Problem

Ivan Contreras[a,c], Jean-François Cordeau[b,c], Gilbert Laporte[*,a,c]

[a] *Canada Research Chair in Distribution Management, HEC Montreal, 3000 chemin de la Côte-Sainte-Catherine, Montreal, Canada H3T 2A7*
[b] *Canada Research Chair in Logistics and Transportation, HEC Montreal, 3000 chemin de la Côte-Sainte-Catherine, Montreal, Canada H3T 2A7*
[c] *Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Montreal, Canada*

Abstract

This paper presents a dynamic (or multi-period) *hub location problem*. It proposes a branch-and-bound algorithm that uses a Lagrangean relaxation to obtain lower and upper bounds at the nodes of the tree. The Lagrangean function exploits the structure of the problem and can be decomposed into smaller subproblems which can be solved efficiently. In addition, some reduction procedures based on the Lagrangean bounds are implemented. These yield a considerable reduction of the size of the problem and thus, help reduce the computational burden. Numerical results on a battery of instances with up to 100 nodes and 10 time periods are reported.

Key words: dynamic hub location, Lagrangean relaxation, branch–and–bound

1. Introduction

Hub Location Problems (HLPs) lie at the heart of network design planning in transportation systems, namely in the airline and trucking industries. The performance of these systems can be improved by using transshipment points, usually called hubs, where the flows between O/D pairs are consolidated and rerouted to their destinations, sometimes via another hub. Thus,

*Corresponding author
Email addresses: ivan.contreras@cirrelt.ca (Ivan Contreras), jean-francois.cordeau@hec.ca (Jean-François Cordeau), gilbert.laporte@cirrelt.ca (Gilbert Laporte)

the locations of the hubs as well as the paths for sending the flows between the origin-destination pairs have to be determined. There are two assumptions underlying most HLPs. The first is that all flows have to be consolidated by hubs. Thus, the paths between O/D pairs must include at least one hub node. The second is that it is possible to fully interconnect hubs with more effective, higher volume pathways that allow a discount factor α ($0 < \alpha < 1$) to be applied to the transportation cost of the flows between any pair of hubs. Broadly speaking, HLPs consist of locating hubs on a network so as to minimize the total flow cost, subject to the above assumptions.

Due to their multiple applications these problems have recently received increasing attention. There exist several variants of HLPs, which differ according to various assumptions regarding the number of hubs to be located, the way the O/D points are assigned to hubs, the existence of limited capacity on the hubs, the way hubs are interconnected, or the independence of the flow discounted costs on all inter-hub links. See Alumur and Kara (2008) and Campbell et al. (2002) for recent surveys.

One common feature of real applications is the dynamic nature of the problem. Scenarios (costs, demand, resources, etc.) often vary over the planning horizon. From the location point of view this gives rise to different types of multi-period, or dynamic, problems. In classical discrete location, dynamic location models have been investigated since the early works of Warszawski (1973) and Van Roy and Erlenkotter (1982), until the more recent contributions of Drezner (1995), Current et al. (1998), Melo et al. (2006), and Albareda-Sambola et al. (2008), among others. Most of these papers have studied the design of supply structures by deciding when and where to locate facilities and when these facilities should be closed. In this type of problems, not only an assignment (or transportation) plan has to be made, but the times at which facilities are opened or closed must be determined.

To the best of the authors' knowledge, there is only one published paper related to dynamic hub location. Campbell (1990) develops a continuous approximation model to locate transportation terminals (hubs) for a general freight carrier serving an increasing demand in a fixed region. It can be seen as a continuous dynamic hub location model in which it is assumed that the origin and destination points of shipments are scattered randomly over the service region. The author investigates the performance of myopic location

strategies based only on the current demands and terminal locations, not on the forecasts of future demand patterns. These myopic terminal location strategies, which are analyzed in both one and two dimensions, provide upper and lower bounds on the transportation cost for the (unknown) optimal strategy.

In this paper we introduce the *Dynamic Uncapacitated Hub Location Problem* (DUHLP) which consists in minimizing the total cost over a finite time planning horizon while ensuring that at each single period all demand is fully routed through the network. The costs include those for the location, operation and closing of hub facilities over time, and the costs of routing the flow through the network. We assume that the demand between O/D pairs varies over the time horizon. Moreover, we allow hub facilities to be opened and closed at different time periods to provide a flexible hub network. The problem is clearly NP-hard since it reduces to the classical *Uncapacitated Hub Location Problem* (UHLP) (O'Kelly, 1992), when the planning horizon consists of a single period.

One of the main difficulties in solving HLPs is the huge number of variables and constraints needed to model them. For this reason, formulations with fewer variables and constraints should be preferred. However, very frequently larger formulations lead to tighter bounds associated with their LP relaxations. This is the case for the UHLP (see for instance, Campbell, 1994; Skorin-Kapov et al., 1997; Ernst and Krishnamoorthy, 1998) where formulations based on four-index variables (or path variables) are much tighter than formulations based on three-index variables (or flow variables). In fact, the recently improved path-based formulations (Hamacher et al., 2004; Cánovas et al., 2006; Marín, 2005) only uses facet defining inequalities. Nevertheless, the main disadvantages of these path-based formulations are the considerable increase in CPU times and memory requirements. We must therefore resort to decomposition techniques to handle these type of formulations.

We propose a quadratic integer programming formulation for the DUHLP based on the formulation of Hamacher et al. (2004) for the UHLP. This formulation has a quadratic objective function with linear constraints. In order to solve it, we propose a Lagrangean relaxation procedure that relaxes the linking constraints of the location and routing variables. As a consequence, we obtain a Lagrangean function that can be decomposed into two indepen-

dent subproblems which can be solved efficiently.

The remainder of this paper is organized as follows. Section 2 formally defines the problem and presents a mathematical programming formulation for the DUHLP. Section 3 describes the proposed Lagrangean relaxation, and analyzes the structure of the subproblems and their solutions. Section 4 theoretically compares the LP bounds obtained with the quadratic formulation and a linearized formulation of the problem. The reduction tests are presented in Section 5. Section 6 presents the exact algorithms that we have developed. The description of the computational results and the analysis of the obtained results are given in Section 7, followed by conclusions in Section 8.

2. Formal Definition and Formulation of the Problem

Let H be a set of potential hub locations, T a set of time periods in the considered time horizon, and $K_1, \ldots, K_{|T|}$ the sets of commodities for each time period. Let W_k^t denote the amount of commodity k to be transported at period t. For each node $i \in H$, f_i^t denotes the fixed cost of opening a hub at node i at the beginning of period t, g_i^t denotes the cost of operating a hub at node i in period t, and q_i^t denotes the recovery gain associated with closing a hub located at node i at the beginning of period t. The distance between nodes i and j in period t satisfies the triangle inequality. These distances are weighted by a discount factor α between two hub nodes to represent the economies of scale obtained by consolidating flows. The DUHLP consists in selecting a set of hubs to be established and the routing of flow through the network at every time period, with the objective of minimizing the sum of net fixed and operational hub costs, plus the transportation costs.

Given that hubs are fully interconnected and the triangle inequality holds, O/D paths will always contain at least one and at most two hub nodes. For this reason, all paths are of the form $(o(k), i, j, d(k))$, where $o(k)$ and $d(k)$ represent the origin and destination nodes of commodity k, respectively, and $(i, j) \in H \times H$ is the ordered pair of hubs (not necessarily distinct) to which $o(k)$ and $d(k)$ are allocated, respectively. Therefore, the transportation cost of routing commodity k along the path $(o(k), i, j, d(k))$ at period t is given by

$$\widehat{F}^t_{ijk} = W^t_k \left(d^t_{o(k)i} + \alpha d^t_{ij} + d^t_{jd(k)} \right).$$

However, given that in any optimal solution every commodity will use at most one direction of a hub edge $e = (e_1, e_2) \in H \times H$ (the one with cheaper transportation cost) we can define undirected transportation costs (see Hamacher et al., 2004). Hence, let $E = \{L \subseteq H : 1 \leq |L| \leq 2\}$ be a set of subsets of H containing one or two hubs. We define the undirected transportation cost F^t_{ek} for each $e \in E$, $k \in K^t$ and $t \in T$ as

$$F^t_{ek} \in \min \left\{ \widehat{F}^t_{ijk}, \widehat{F}^t_{jik} \right\}.$$

In what follows, we present an integer quadratic programming formulation for the DUHLP based on the formulation proposed by Hamacher et al. (2004) for the UHLP. For each $k \in K^t$, $e \in E$ and $t \in T$ we define

$$x^t_{ek} = \begin{cases} 1 & \text{if commodity } k \text{ at period } t \text{ uses hub edge } e; \\ 0 & \text{otherwise.} \end{cases}$$

For modeling the location of hubs we define

$$z^t_i = \begin{cases} 1 & \text{if a hub facility is located at node } i \text{ in period } t; \\ 0 & \text{otherwise} \end{cases}$$

for each $i \in H$ and $t \in T$. Using these two sets of variables, the DUHLP can be stated as

$$\begin{aligned}
\text{(QM) minimize} \quad & \sum_{i \in H} \sum_{t \in T} f^t_i (1 - z^{t-1}_i) z^t_i + \sum_{i \in H} \sum_{t \in T} g^t_i z^t_i \\
& - \sum_{i \in H} \sum_{t \in T} q^t_i (1 - z^t_i) z^{t-1}_i + \sum_{e \in E} \sum_{k \in K^t} \sum_{t \in T} F^t_{ek} x^t_{ek} \\
\text{subject to} \quad & \sum_{e \in E} x^t_{ek} = 1 && k \in K^t, t \in T && (1) \\
& \sum_{\{e \in E \,:\, i \in e\}} x^t_{ek} \leq z^t_i && i \in H, k \in K^t, t \in T && (2) \\
& z^t_i \in \{0, 1\} && i \in H, t \in T && (3) \\
& x^t_{ek} \geq 0 && e \in E, k \in K^t, t \in T. && (4)
\end{aligned}$$

The first term of the objective function represents the opening cost of the hub facilities. In particular, an opening cost is incurred at node i in time

period t if there exists an open hub at i in period t and if there is no open hub at i in period $t-1$. The second term represents the total operational cost, whereas the third term represents the total recovery gain generated by closing hub facilities. In particular, there is a recovery gain at node i in time period t if there exists an open hub at i in period $t-1$ and if there is no open hub at i in period t. The last term represents the total routing cost. The constraints are almost the same as in the UHLP except that here, they depend on the time period. Constraints (1) guarantee that for each commodity there exists a single path connecting its origin and destination nodes at each time period t. Constraints (2) prohibit commodities from being routed via a node that is not a hub, at each time period t. Finally, constraints (3) and (4) are the usual integrality and non-negativity constraints.

3. Lagrangean Relaxation

Lagrangean Relaxation (LR) is a well known method for solving large-scale combinatorial optimization problems. It exploits the inherent structure of the problems to compute lower bounds on the value of the optimal solution. In the case of model QM, if we relax constraints (2) in a Lagrangean fashion, weighting their violations with a multiplier vector u of appropriate dimension, we obtain the following Lagrangean function:

$$\begin{aligned} L(u) = \text{minimize} \quad & \sum_{i\in H}\sum_{t\in T} f_i^t(1-z_i^{t-1})z_i^t + \sum_{i\in H}\sum_{t\in T} g_i^t z_i^t \\ & - \sum_{i\in H}\sum_{t\in T} q_i^t(1-z_i^t)z_i^{t-1} + \sum_{e\in E}\sum_{k\in K^t}\sum_{t\in T} F_{ek}^t x_{ek}^t \\ & + \sum_{i\in H}\sum_{k\in K^t}\sum_{t\in T} u_{ik}^t \left(\sum_{\{e\in E\,:\,i\in e\}} x_{ek}^t - z_i^t \right) \\ \text{subject to} \quad & (1), (3), (4). \end{aligned}$$

Note that $L(u)$ is separable into two subproblems: 1) a problem in the space of the z variables, and 2) a problem in the space of the x variables. After some algebra, the first subproblem can be expressed as

$$\begin{aligned} L_z(u) = \text{minimize} \quad & \sum_{i\in H}\sum_{t\in T} \left[\left(f_i^t + g_i^t - q_i^{t+1} - \sum_{k\in K^t} u_{ik}^t \right) z_i^t - \left(f_i^t - q_i^t \right) z_i^t z_i^{t-1} \right] \\ \text{subject to} \quad & (3), \end{aligned}$$

and the second subproblem can be expressed as

$$\begin{aligned} L_x(u) = \text{minimize} \quad & \sum_{e\in E}\sum_{k\in K^t}\sum_{t\in T} \overline{F}^t_{ek} x^t_{ek} \\ \text{subject to} \quad & (1), (4), \end{aligned}$$

where the coefficients of the objective function are:

- $\overline{F}^t_{ek} = \begin{cases} F^t_{ek} + u_{e_1k} + u_{e_2k}, & \text{if } |e| = 2 \\ F^t_{ek} + u_{e_1k}, & \text{if } |e| = 1. \end{cases}$

Therefore, we obtain the following result:

Proposition 1.

$$L(u) = L_z(u) + L_x(u).$$

3.1. ***Solution to subproblem*** $L_z(u)$

Observe that subproblem $L_z(u)$ can be decomposed into $|H|$ independent problems, one for each candidate hub node $i \in H$, of the form:

$$\begin{aligned} (QAD_i) \quad Q^*_i(u) = \text{minimize} \quad & Q_i = \sum_{t\in T} \left(\alpha^t_i z^t_i - \beta^t_i z^t_i z^{t-1}_i\right) \\ \text{subject to} \quad & z^t_i \in \{0,1\} \qquad\qquad t \in T, \end{aligned} \tag{5}$$

where the coefficients of the objective function are

- $\alpha^t_i = \left(f^t_i + g^t_i - q^{t+1}_i - \sum_{k\in K^t} u^t_{ik}\right)$
- $\beta^t_i = \left(f^t_i - q^t_i\right).$

Each subproblem QAD_i can be efficiently solved to optimality by using dynamic programming. Let Q^s_i denote the expression of Q_i for the first s time periods of the dynamic program, $Q^s_i(0)$ the best value of Q^s_i if $z^t_i = 0$, and $Q^s_i(1)$ the best value if $z^t_i = 1$. Then, for $s = 1$,

$$Q^1_i \;\; = \;\; \alpha^1_i z^1_i - \beta^1_i z^1_i z^0_i$$

so that

- $Q_i^1(0) = 0$
- $Q_i^1(1) = \alpha_i^1 - \beta_i^1 z_i^0$

and for $1 < s \leq |T|$,

$$\begin{aligned} Q_i^{s+1} &= \sum_{t=1\ldots s+1} \left(\alpha_i^t z_i^t - \beta_i^t z_i^t z_i^{t-1}\right) \\ &= Q_i^s + \alpha_i^{s+1} z_i^{s+1} - \beta_i^{s+1} z_i^{s+1} z_i^s \end{aligned}$$

so that,

- $Q_i^{s+1}(0) = \min\left\{Q_i^s(0), Q_i^s(1)\right\}$
- $Q_i^{s+1}(1) = \min\left\{Q_i^s(0), Q_i^s(1) - \beta_i^{s+1}\right\} + \alpha_i^{s+1}$.

Finally,

$$Q_i^*(u) = \min\left\{Q_i^{|T|}(0), Q_i^{|T|}(1)\right\}$$

and therefore,

$$L_z(u) = \sum_{i \in H} Q_i^*(u) = \sum_{i \in H} \min\left\{Q_i^{|T|}(0), Q_i^{|T|}(1)\right\}.$$

For a given candidate hub node $i \in H$, the time required to solve QAD_i by dynamic programming is dominated by that needed to obtain the coefficients α_i^t, $t \in T$. For each t, α_i^t can be computed in $\mathcal{O}(|K^t|)$ operations. Then, solving each QAD_i by dynamic programming has a complexity of $\mathcal{O}(\sum_{t \in T} |K^t|)$. Thus, the overall complexity for solving $L_z(u)$ is $\mathcal{O}(\sum_{t \in T} |K^t| \cdot |H|)$.

3.2. ***Solution to subproblem*** $L_x(u)$

Subproblem $L_x(u)$ is a semi-assignment problem which, in turn, can be decomposed into $\sum_{t \in T} |K^t|$ independent semi-assignment problems, corresponding to each commodity $k \in K^t$ and to each period t, of the form:

$$(SAP_k^t)\ q_k^t(u) = \text{minimize} \quad \sum_{e\in E} \overline{F}^t_{ek} x^t_{ek}$$

$$\text{subject to} \quad \sum_{e\in E} x^t_{ek} = 1 \tag{6}$$

$$x^t_{ek} \geq 0 \qquad e \in E. \tag{7}$$

For a given $k \in K^t$ and a given t, the optimal value to SAP_k^t is

$$q_k^t(u) = \overline{F}^t_{\widehat{e}k},$$

where $\widehat{e} \in E$ is arbitrarily selected such that

$$\overline{F}^t_{\widehat{e}k} = \min\left\{\overline{F}^t_{ek} : e \in E\right\}. \tag{8}$$

Therefore,

$$L_x(u) = \sum_{k\in K^t}\sum_{t\in T} q_k^t(u) = \sum_{k\in K^t}\sum_{t\in T} \min\left\{\overline{F}^t_{ek} : e \in E\right\}.$$

For a given commodity $k \in K^t$ and a given t, evaluating the closed expression (8) has a complexity of $\mathcal{O}(|E|)$. Thus, the overall complexity for solving $L_x(u)$ is $\mathcal{O}(\sum_{t\in T} |K^t| \cdot |E|)$.

3.3. ***The solution of the Lagrangean Dual***

Proposition 2. *For a given vector of multipliers* (u) *the Lagrangean function* $L(u)$ *can be solved in* $\mathcal{O}(\sum_{t\in T} |K^t| \cdot |E|)$ *time.*

Proof Solving $L_y(u)$ has complexity $\mathcal{O}(\sum_{t\in T} |K^t| \cdot |H|)$, whereas solving $L_x(u)$ has complexity $\mathcal{O}(\sum_{t\in T} |K^t| \cdot |E|)$. Given that $|H| < |E|$, and because of Proposition 1, the result follows. ■

In order to obtain the best lower bound one must solve the Lagrangean dual of QM, which is given by

$$(D) \quad z_D = \max_{u\geq 0} \quad L(u). \tag{9}$$

We apply subgradient optimization to solve problem D. It is well known that classical subgradient algorithms tend to suffer from slow convergence when solving the Lagrangean dual problem. We therefore propose a deflected subgradient method (see Camerini et al., 1975) to improve the convergence of the algorithm. Whereas the classical subgradient algorithm only uses the subgradient of the current iteration to compute the direction of movement, that is $d^k = s^k$, deflected subgradient algorithms take into account the direction of the previous iteration to obtain the current direction, that is $d^k = s^k + \theta^k d^{k-1}$. The effectiveness of these methods relies on the choice of the deflection parameter θ^k and several attempts have been described to provide clever choices of this parameter (for instance, Crowder, 1976; Sherali and Ulular, 1989; Camerini et al., 1975; Brännlund, 1995). We use the following rule based on geometrical arguments (see Camerini et al., 1975):

$$\theta^k = \begin{cases} \left\|s^k\right\| / \left\|d^{k-1}\right\| & \text{if } s^k d^{k-1} < 0, \\ 0 & \text{otherwise.} \end{cases}$$

Now, for a given vector (u), let $z(u)$, and $x(u)$ denote the optimal solution to $L(u)$. Then, a subgradient of $L(u)$ is given by

$$\gamma(u) = \left(\left(\sum_{\{e \in E \,:\, i \in e\}} x^t_{ek} - z^t_i \right)_{i,k,t} \right).$$

An implementation of the deflected subgradient algorithm is depicted in Algorithm 1. The output of the algorithm is a lower bound z_D, and $\bar{\eta}$ denotes a known upper bound on the optimal value of the original problem. The parameter λ^k is halved after 25 consecutive iterations without improvement in the lower bound and is reset to 2 every 200 iterations.

Algorithm 1 Deflected Subgradient Method

Iteration 0

Initialize $z_D = -\infty$; $u^0 = 0$; $d^0 = 0$; $\lambda^k = 2$.

Let $\bar{\eta}$ be a known upper bound on the optimal solution value.

Iteration k

Solve the lagrangean function $L(u^k)$.

if $\left(L(u^k) > z_D\right)$ **then**

$z_D \leftarrow L(u^k)$

end if

Evaluate the subgradient $\gamma(u^k)$.

if $\left(\gamma(u^k)d^{k-1} < 0\right)$ **then**

$\theta^k = \left\|\gamma(u^k)\right\| / \left\|d^{k-1}\right\|$

else

$\theta^k = 0$

end if

Obtain the direction $d^k = \gamma(u^k) + \theta^k d^{k-1}$.

Calculate the step length $t^k \leftarrow \lambda^k \frac{\left(\bar{\eta} - L(u^k)\right)}{\gamma(u^k)d^{k-1}}$.

Set $\left(u^{k+1}\right) \leftarrow \left(u^k\right) + t^k d^k$.

Set $k \leftarrow k + 1$.

3.4. ***Upper bounds from primal solutions***

We can exploit the primal information obtained from the Lagrangean function to construct feasible solutions. Let $\widehat{H}^t(u) = \{i : z_i^t(u) = 1, i \in H\}$ and $\widehat{E}^t(u) = \{L \subseteq R^t(u) : 1 \leq |L| \leq 2\}$ be the current set of open hub facilities and available hub edges at period t, respectively, associated with the primal solution $z(u)$ obtained by solving $L_z(u)$. In the case of the UHLP, when the location of the hub facilities is known, we can solve the routing problem of the commodities very efficiently. The same situation occurs in the case of the DUHLP because once we know the location of the hub facilities in the entire planning horizon, we can decompose the problem into independent routing problems, one for each time period $t \in T$.

In order to construct a feasible solution we only need to ensure that there exists at least one hub node at each time period $t \in T$, that is $\sum_{i \in H} z_i^t(u) \geq 1$. If this is the case at a given primal solution $z(u)$, we can compute an upper bound for QM as

$$\bar{\eta}(u) = \sum_{i\in H}\sum_{t\in T}\left[\left(f_i^t + g_i^t - q_i^{t+1}\right) z_i^t(u) - \left(f_i^t - q_i^t\right) z_i^t(u) z_i^{t-1}(u)\right] + \sum_{k\in K^t}\sum_{t\in T} \min\left\{F_{ek}^t : e \in \widehat{E}^t(u)\right\}.$$

4. Comparison of Bounds

Consider the following linear mixed integer programming problem obtained by linearizing model QM in the classical way:

$$\begin{aligned}
\text{(LM) minimize} \quad & \sum_{i\in H}\sum_{t\in T}(f_i^t + g_i^t - q_i^{t+1})z_i^t - \sum_{i\in H}\sum_{t\in T}(f_i^t - q_i^t)y_i^t \\
& + \sum_{e\in E}\sum_{k\in K^t}\sum_{t\in T} F_{ek}^t x_{ek}^t \\
\text{subject to} \quad & (1)-(4) \\
& y_i^t \le z_i^{t-1} \qquad i \in H, t \in T \qquad (10) \\
& y_i^t \le z_i^t \qquad i \in H, t \in T \qquad (11) \\
& y_i^t \in \{0,1\} \qquad i \in H, t \in T. \qquad (12)
\end{aligned}$$

Observe that the usual constraints $z_i^t + z_i^{t-1} - y_i^t \le 1$ for each $i \in H$ and $t \in T$ are not necessary because of the non-positive coefficient of y_i^t variables and the minimization of the objective function. Now, the following result states the equivalence of the optimal solution value of the LP relaxations of models LM and QM.

Proposition 3. *The value of the LP relaxation of model LM coincides with the value of the dual problem D of model QM.*

Proof The Lagrangean function can be stated as

$$\begin{aligned}
L(u) = L_y(u) + L_x(u) = \text{ minimize} \quad & \sum_{i\in H}\sum_{t\in T}\left(\alpha_i^t z_i^t - \beta_i^t z_i^t z_i^{t-1}\right) \\
& + \sum_{k\in K^t}\sum_{t\in T} q_k^t(u) \\
\text{subject to} \quad & (3).
\end{aligned}$$

Following the scheme of the proof given in Billionnet and Elloumi (1992) and Chardaire et al. (1996), as $\beta_i^t \geq 0$, $i \in H$ and $t \in T$, calculating the Lagrangean function amounts to minimizing a quadratic pseudo-Boolean submodular function. This can be done solving a continuous linear program (see Rhys, 1970). Let us now consider a linearization of the Lagrangean function

$$
\begin{aligned}
L(u) = \text{ minimize } & \sum_{i \in H} \sum_{t \in T} \left(\alpha_i^t z_i^t - \beta_i^t y_i^t\right) + \sum_{k \in K^t} \sum_{t \in T} q_k^t(u) & \\
\text{subject to } & y_i^t \leq z_i^{t-1} \qquad i \in H, t \in T & (13)\\
& y_i^t \leq z_i^t \qquad i \in H, t \in T & (14)\\
& z_i^t, y_i^t \in \{0,1\} \qquad i \in H, t \in T. & (15)
\end{aligned}
$$

Since (13)–(14) define a totally unimodular matrix, we can remove the integrality constraints (15) and replace them with $0 \leq z_i^t \leq 1$ and $0 \leq y_i^t \leq 1$, for all $i \in H$, and all $t \in T$. The Lagrangean function is equal to the solution of the associated continuous problem. Thus, the dual problem D can be considered as the Lagrangean dual problem of the continuous relaxation of LM obtained by relaxing constraints (2). Therefore, the optimal solution value of the dual problem D is equal to that of the continuous relaxation of LM. ■

5. Reduction Tests

As mentioned earlier, one of the main drawbacks of model QM is its very large number of variables and constraints, even for small size instances. In the previous sections, we have described a Lagrangean relaxation approach that is able to handle this huge number of constraints and to solve in polynomial time the resulting Lagrangean function. Unfortunately, as the size of the instances increases, the number of variables in the Lagrangean function is so large that it takes a considerable amount of computational time to evaluate this function.

One way of reducing the size of the model is to develop some reduction tests capable of eliminating variables for which it is known that either they do not appear, or they take value one in at least one optimal solution. In this section, we develop simple reduction tests capable of determining if, in a given time period, a hub will be open or closed in an optimal solution of a given instance. Similar reduction tests have been successfully applied

for other HLPs (see Contreras et al., 2009). Both tests use the information obtained from the Lagrangean function at a given iteration to estimate the location and traffic costs, in case that a node is chosen to become a hub or not. The idea of the tests is to consider subsets $S_i \subseteq H$ of candidate time periods for each $i \in H$, while ensuring that the bounds derived from the Lagrangean function are still valid. To this end, for every $i \in H$ we partition the set of time periods T into $S \cup S_i^0 \cup S_i^1$, where S_i^0 and S_i^1, are the sets of indices of time periods that take value 0 and 1, respectively, in the optimal solution of the problem. Let $S = \{S_i\}_{i\in H}$, $S^0 = \{S_i^0\}_{i\in H}$, $S^1 = \{S_i^1\}_{i\in H}$ be the set of all subsets of indices of time periods that are not fixed, that take value 0 and 1, respectively. For each $t \in T$, let $R^t = \{i : t \in S_i \cup S_i^1, i \in H\}$ be the set of candidate hub nodes at period t. Given a multiplier vector (u), a node $j \in H$, and a time period $r \in S_j$, we define:

$$\begin{aligned}\Delta_{rj}^1(u, S^1, S^0, S) &= \widehat{Q}_j^*(u, S_j^1 \cup r, S_j^0, S_j \setminus r) + \sum_{i\in H\setminus\{j\}} \widehat{Q}_i^*(u, S_i^1, S_i^0, S_i) \\ &\quad + \sum_{k\in K^t} \sum_{t\in T} q_k^t(u, R^t)\end{aligned}$$

and

$$\begin{aligned}\Delta_{rj}^0(u, S^1, S^0, S) &= \widehat{Q}_j^*(u, S_j^1, S_j^0 \cup r, S_j \setminus r) + \sum_{i\in H\setminus\{j\}} \widehat{Q}_i^*(u, S_i^1, S_i^0, S_i) \\ &\quad + \sum_{k\in K^t} \sum_{t\in T\setminus\{r\}} q_k^t(u, R^t) + \sum_{k\in K^t} q_k^r(u, R^t \setminus \{j\})\end{aligned}$$

where

$$\begin{aligned}\widehat{Q}_i^*(u, O, C, U) = \text{minimize} \quad & \sum_{t\in T} \left(\alpha_i^t z_i^t - \beta_i^t z_i^t z_i^{t-1}\right) \\ \text{subject to} \quad & z_i^t = 0 \qquad t \in C \\ & z_i^t = 1 \qquad t \in O \\ & z_i^t \in \{0,1\} \qquad t \in U,\end{aligned}$$

and

$$\hat{q}_k^t(u, A) = \min\left\{\overline{F}_{ek}^t : e \in \{L \subseteq A : 1 \le |L| \le 2\}\right\}.$$

Observe that $\Delta^1_{rj}(u, S^1, S^0, S)$ is the value of $L(u)$ when *i)* we locate a hub at node j in time r, *ii)* a hub is located at node j in each $t \in S^1_j$, *iii)* in $L_z(u)$ at node j we restrict the set of additional candidate time periods to $S_j \setminus \{r\}$ and, *iv)* in $L_x(u)$ we restrict the hub edges to R^t for every $t \in T$. Similarly, $\Delta^0_{rj}(u, S^1, S^0, S)$ is the value of $L(u)$ when *i)* a hub is located at node j in each $t \in S^1_j$, *ii)* we restrict the set of additional candidate time periods to $S_j \setminus \{r\}$ for node j, *iii)* in $L_x(u)$ we restrict the candidate hub nodes to R^t for every $t \in T \setminus \{r\}$ and , *iv)* for period r we restrict the candidate hub nodes to $R^t \setminus \{j\}$. The following result gives reduction tests for fixing the status of hubs.

Proposition 4. *Let $\bar{\eta}$ be an upper bound on the optimal solution value η^* and (u) a multipliers vector. For each $i \in H$, let $T = S \cup S^0_i \cup S^1_i$ be a partition such that a hub is located at node i at any time period of S^1_i in any optimal solution and, for each $t \in T$, R^t contains the optimal set of hubs for period t. Then*

- *If there exists $j \in H$ and $r \in S_j$ such that $\Delta^1_{rj}(u, S^1, S^0, S) > \bar{\eta}$, then $z^r_j = 0$ in any optimal solution.*
- *If there exists $j \in H$ and $r \in S_j$ such that $\Delta^0_{rj}(u, S^1, S^0, S) > \bar{\eta}$, then $z^r_j = 1$ in any optimal solution.*

Proof

- Given that S_j is such that $S_j \cup S^1_j$ contains the optimal set of hubs, $\Delta^1_{rj}(u, S^1, S^0, S)$ is a lower bound on the objective function value if a hub is located at node j in time period r. Therefore, if $\Delta^1_{rj}(u, S^1, S^0, S) > \bar{\eta}$, it holds that $z^r_j = 0$ in any optimal solution.
- Similarly, $\Delta^0_{rj}(u, S^1, S^0, S)$ is a lower bound on the objective function value if no hub is located at node j in time period r. Therefore, if $\Delta^0_{rj}(u, S^1, S^0, S) > \bar{\eta}$, it holds that $z^r_j = 1$ in any optimal solution.

■

In our algorithmic framework, the above result is used in the following way. At the beginning of the subgradient optimization we consider all time periods as candidates for each $i \in H$, that is $S_i = T$, $S^0_i = S^1_i = \emptyset$. Then, everytime we obtain an improved lower bound, we apply the reduction tests

for every $i \in H$ and $r \in S_i$. When a particular z_j^r variable is fixed, the set S_j is updated to $S_j \setminus \{r\}$. In addition, when we fix $z_j^r = 1$, we update $S_i^1 := S_i^1 \cup \{r\}$, whereas when we fix $z_j^r = 0$, we update $S_i^0 := S_i^0 \cup \{r\}$. Finally, we update $R^t := R^t \setminus \{j\}$. Proposition 4 guarantees that in both cases the updated set R^t always contains the optimal set of hubs for each time period t.

When the sets of potential time periods reduce to $S_i \subsetneq T$ for $i \in H$, the expression of the Lagrangean function is updated accordingly to $L(u, S, S^1, S^0)$. Observe that, since for each variable that is fixed there is an important reduction on the number of variables, the evaluation of $L_x(u, S, S^1, S^0)$ can be performed more efficiently. In fact, the complexity for solving the updated Lagrangean function is $\mathcal{O}(\sum_{t \in T} |K^t| \cdot |A^t|)$, where $A^t = \{L \subseteq R^t : 1 \leq |L| \leq 2\}$.

6. Proving Optimality

As we will see in the computational experiments section, for some of the test instances the Lagrangean relaxation algorithm enables us to prove the optimality of the best found solution. When optimality cannot be proven, most often the duality gap is very small. However, our goal is to propose an exact algorithm that proves the optimality of the obtained solutions. Thus, when the Lagrangean relaxation algorithm does not prove optimality of the best solution found we must resort to an enumeration algorithm. In this section we describe two different enumeration approaches for solving the problem to optimality. The first one is a standard branch-and-bound method in which at every node of the branching tree we obtain lower and upper bounds by using the Lagrangean relaxation algorithm. The second one includes a partial enumeration phase which enhances the application of the reduction tests. It is applied at the beginning of the branch-and-bound procedure so as to reduce the number of variables to branch on, and the sizes of the subproblems in the nodes of the branching tree.

Let $S = \{S_i\}_{i \in H}$, $R^t = \{i : t \in S_i \cup S_i^1, i \in H\}$, and $(\hat{z}, \hat{x})$ denote the set of sets of candidate time periods for each node $i \in H$, the set of candidate hub nodes at period t, and the best solution found at the end of the Lagrangean relaxation algorithm, respectively.

6.1. ***Standard branch-and-bound***

As mentioned, the proposed Lagrangean relaxation can be incorporated in a branch-and-bound algorithm in order to obtain the optimal solution of the problem and verify it. Thus, Algorithm 3.3 is used as a bounding procedure at every node of the enumeration tree to produce both lower and upper bounds.

The branching strategy that we have used works as follows. If there are any unfixed z_i^t variables such that $\hat{z}_i^t = 1$, we identify among these the one with the largest reduced cost, i.e. α_i^t, and explore the 1-branch. This way, the algorithm starts with the branch that seems to be optimal and leaves for later the branch that seems to be non-optimal (the 0-branch).When there are no more unfixed z_i^t variable such that $\hat{z}_i^t = 1$, we branch on the remaining unfixed variables by identifying again the one with the largest reduced cost and explore the 1-branch. Now, the algorithm starts with the branch that seems to be non-optimal and leaves the 0-branch for later.

We explore the branch-and-bound tree in a depth first search fashion. Instead of starting from scratch when solving the dual problem associated to a particular node, dual solutions from its parent node often provide a natural and very good starting point for the subgradient method. In fact, we set the number of subgradient iterations to 200 for all nodes of the branching tree (except at the root node where it is set to 1500).

Given that our solution method in the nodes of the search tree is approximate, since branching is made when the subgradient method stops but does not necessarily converge to the optimal solution, in many cases unnecessary branching is performed. Therefore, we can expect the branch-and-bound tree to be larger than it would be for a branch-and-bound procedure based on the LP relaxation. However, as we will see in the computation experiments section, we are able to obtain much more efficiently the lower bound of the Lagrangean relaxation than that of the LP relaxation. Thus, this could compensate for the increase of the tree.

6.2. ***Branch and bound with partial enumeration***

The partial enumeration works as follows. For each $i \in H$ and $t \in S_i$, we temporarily fix $z_i^t = 1$ and solve the resulting Lagrangean problem

$L(u, S \setminus t, S^1 \cup t, S^0)$. If the resulting lower bound lb_{it}^1 is greater than the current best upper bound, we set $z_i^t = 0$ and the related x variables, and we update the sets S_i, S_i^0 and R^t accordingly. Otherwise, we temporarily fix $z_i^t = 0$ (but only if $\hat{z}_i^t = 1$) and solve the resulting Lagrangean $L(u, S \setminus t, S^1, S^0 \cup t)$. If the obtained lower bound lb_{it}^0 is greater than the current best upper bound, we set $z_i^t = 1$, and we update the sets S_i, S_i^0 and R^t accordingly.

If there are still some unfixed variables z_i^t at the the end of the partial enumeration phase we continue with the branch-and-bound scheme just described. However, the branching strategy is determined by the output of the partial enumeration phase. For each element in S_i and $i \in H$, we consider $\delta_i^t = \max\{lb_{it}^0, lb_{it}^1\}$, and we branch on the variable z_j^r such that $\delta_j^r = \max\{\delta_i^t : t \in S_i, i \in H\}$. Then, if $\delta_j^r = lb_{rj}^0$, we explore the branch corresponding to $z_j^r = 1$; otherwise, we explore the branch corresponding to $z_j^r = 0$.

7. Computational Experiments

We have run extensive computational experiments in order to analyze and compare the performance of the formulation, the Lagrangean relaxation, the reduction tests and the exact algorithms. All algorithms were coded in C and run on a Dell Studio PC with an Intel Core 2 Quad processor Q8200 with 2.33 GHz and 8 GB of RAM memory under a Linux environment.

We have used the AP (Australian Post) set of instances which can be downloaded from *mscmga.ms.ic.ac.uk/jeb/orlib/phubinfo.html* to generate a set of benchmark instances for the DUHLP. This data set, which is commonly used in the literature, consists of the Euclidean distances between 200 cities in Australia, a code to reduce the size of the set by grouping cities, and the values of W_k (postal flow between pair of cities). We took a static instance of the AP data set and transformed it to a dynamic instance in the following way. At the first period of the planning horizon ($t = 1$), a subset of commodities K^1 is randomly selected from the set of commodities $K = \{(i, j) : (i, j) \in H \times H\}$ associated to its corresponding static instance. In subsequent time periods $t > 1$, subsets of commodities are randomly selected in such a way that $K^1 \subseteq K^2 \subseteq \cdots \subseteq K^{|T|} = K$. The first time a given commodity appears its demand is set to W_k, otherwise its demand either randomly increases up to 30% of its value (with probability 90%), or

randomly decreases up to 25% of its value (with probability 10%). In the case of the set-up cost, we consider two possibilities: 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. Moreover, we vary set-up costs over time by setting them to $f_i^t = f_i(1+\theta)$, where $\theta \sim U(-0.1, 0.2)$ and f_i is the corresponding set-up cost of the associated static instance. Operating costs are set to $g_i^t = f_i^t\rho$, where $\rho \sim U(0.1, 0.15)$ and recovery gains are set to $q_i^t = f_i^t\delta$, where $\delta \sim U(0.4, 0.6)$. Finally, the different values we have considered for the discount factor are $\alpha = 0.2, 0.5$ and 0.8.

We have considered four sets of instances: the first one contains 30 small to medium size instances with up to 50 nodes and five time periods. This set contains six instances of each of the sizes $|H| = 10, 20, 25, 40$, and 50. The second set contains 30 medium to large size instances with up to 100 nodes and five time periods. In particular, this set contains six instances of each of the sizes $|H| = 60, 70, 75, 90$, and 100. These two sets of instances were generated in such a way that the number of commodities in the different time periods are $|K^1| \approxeq (0.25)|K|$, $|K^2| \approxeq (0.50)|K|$, $|K^3| \approxeq (0.75)|K|$, $|K^4| \approxeq |K|$, and $|K^5| = |K|$. The third set contains 30 instances with up to 50 nodes and 10 time periods. This set contains six instances of each of the sizes $|H| = 10, 20, 25, 40$, and 50. Finally, the fourth set contains 30 instances with up to 100 nodes and 10 time periods. It contains six instances of each of the sizes $|H| = 60, 70, 75, 90$, and 100. The last two sets of instances were generated in such a way that the number of commodities in the different time periods are $|K^1| \approxeq (0.20)|K|$, $|K^2| \approxeq (0.40)|K|$, $|K^3| \approxeq (0.50)|K|$, $|K^4| \approxeq (0.65)|K|$, $|K^5| \approxeq (0.75)|K|$, $|K^6| \approxeq (0.90)|K|$, $|K^7| \approxeq (0.95)|K|$, $|K^8| \approxeq |K|$, and $|K^9| = |K^{10}| = |K|$. For each problem size in each set, the six instances correspond to different combinations of characteristics for the fixed set-up costs and the discount factor α. There are thus 120 test instances in total.

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: $Iter_{max} = 1500$, $t = 350$, $\epsilon = 10^{-6}$, and $\gamma = 0.005\%$.

Our preliminary computational results focus on the comparison between two different Lagrangean relaxations LR and LR_{RT}. LR considers the Lagrangean relaxation without the reduction tests whereas LR_{RT} considers the Lagrangean relaxation with the reduction tests. Moreover, we compare both relaxations with the LP relaxation of the linearized model LM obtained with CPLEX 10.1. The detailed results of these comparisons using the first set of instances are given in Table 1. The first three columns give the number of nodes, the type of fixed cost and the discount factor, respectively, of each instance. The next three columns under the heading *%Deviation* depict the duality gap relative to: *i)* the linear programming relaxation bound (LP) obtained with model LM, *ii)* the best lower bound obtained with LR and *iii)* the best lower bound obtained with LR_{RT}, respectively. That is $Gap = 100(OPT - LB_T)/(OPT)$, where OPT is the optimal value and LB_T is the lower bound obtained with $T = LP, LR, LR_{RT}$, respectively. The next four columns under the heading *Time(sec)* give the CPU time in seconds needed to obtain an optimal solution of the LP relaxation in case of LM and the CPU time in seconds needed to obtain the lower and upper bounds in case of LR and LR_{RT}, respectively. The next column under the heading *% Red Time* gives the percentage of reduction in time of LR_{RT} with respect to LR. Finally, the last column *% Red Hubs* gives the percentage of hubs that where fixed by the reduction tests in LR_{RT}, that is $RED = 100(FH/|H|)$, where FH is the number of hubs fixed.

The results presented in Table 1 appear to be very good. As can be seen we have proven the optimality of the solution obtained with the Lagrangean dual in 12 out of the 30 instances with LR and in 15 out of the 30 instances with LR_{RT}. For the remaining instances, the percent deviation is below 1.2% for both LR and LR_{RT}. Also, it can be seen that the LP bound of model LM is very tight since it is able to close the duality gap in 20 out of the 24 instances with up to 40 nodes. However, for the 50-node instances, eight GB of memory are not enough for loading the problem into the CPLEX solver

			%Deviation			*Time(sec)*			*% Red*	*% Red*
$\|H\|$	*FC*	α	LP	LR	LR_{RT}	LP	LR	LR_{RT}	*time*	*hubs*
10	L	0.2	0.00	0.00	0.00	0.15	0.13	0.04	69.23	77.78
		0.5	0.00	0.00	0.00	0.12	0.05	0.06	-20.00	82.22
		0.8	0.00	0.00	0.00	0.13	0.06	0.06	0.00	82.22
	T	0.2	0.00	0.00	0.00	0.22	0.10	0.05	50.00	82.22
		0.5	0.19	0.20	0.19	0.12	0.34	0.35	-2.94	26.67
		0.8	0.06	0.11	0.13	0.08	0.32	0.23	28.13	35.56
20	L	0.2	0.00	0.00	0.00	8.58	7.24	2.49	65.61	91.58
		0.5	0.00	0.00	0.00	6.03	2.66	1.70	36.09	94.74
		0.8	0.00	0.27	0.10	4.25	4.06	3.64	10.34	51.58
	T	0.2	0.00	0.00	0.00	13.51	5.54	1.36	75.45	91.58
		0.5	0.00	0.00	0.00	6.85	1.94	1.38	28.87	91.58
		0.8	0.06	0.16	0.07	3.45	5.86	3.34	43.00	86.32
25	L	0.2	0.00	0.05	0.00	56.46	15.58	6.21	60.14	92.50
		0.5	0.00	0.28	0.25	30.50	9.57	6.13	35.95	59.17
		0.8	0.00	0.14	0.18	16.65	14.82	8.21	44.60	41.67
	T	0.2	0.00	0.10	0.20	119.01	15.34	8.54	44.33	25.00
		0.5	0.00	0.25	0.29	53.60	17.72	8.76	50.56	78.33
		0.8	0.00	0.17	0.00	25.18	13.21	5.40	59.12	99.17
40	L	0.2	0.00	0.00	0.00	1413.40	76.96	36.41	52.69	95.90
		0.5	0.07	0.45	0.57	804.56	77.11	47.59	38.28	41.03
		0.8	0.00	0.44	0.33	426.49	114.05	81.88	28.21	46.15
	T	0.2	0.00	0.27	0.00	2190.62	102.77	46.89	54.37	98.97
		0.5	0.00	0.00	0.00	702.06	94.54	21.87	76.87	98.97
		0.8	0.00	0.00	0.00	279.77	105.17	21.33	79.72	98.97
50	L	0.2	n.a.	0.51	0.41	n.a.	278.53	187.89	32.54	53.06
		0.5	n.a.	0.22	0.11	n.a.	279.10	184.40	33.93	77.14
		0.8	n.a.	0.40	0.40	n.a.	279.52	253.40	9.34	24.08
	T	0.2	n.a.	0.05	0.00	n.a.	277.11	75.02	72.93	97.96
		0.5	n.a.	1.07	0.97	n.a.	250.45	141.83	43.37	44.90
		0.8	n.a.	1.17	1.17	n.a.	276.34	138.95	49.72	35.51

Table 1: Comparison of bounds for instances from 10 to 50 nodes and five time periods

optimizer. In contrast, our Lagrangean relaxation is able to obtain tight bounds for the 50 node instances by using less than one GB of memory.

The columns *Time(sec)* indicate that, particularly for LR_{RT}, the Lagrangean relaxation requires much less computation time than CPLEX. Except for two 10-node instances, our Lagrangean relaxation is always much faster than CPLEX. For the case of the 40-node instances, the Lagrangean relaxation is at least one order of magnitude faster than CPLEX. For the case of the 50-nodes instances, the Lagrangean relaxation is able to compute really tight bounds in less than five minutes. The last two columns show that the proposed reduction tests are very effective in fixing hub nodes. The percentage of hubs fixed ranges from 25% to 99% and the average percentage is 70%. As can be seen in the *% Red Time* column, LR_{RT} clearly outperforms LR because it is able to considerably reduce the computational effort in all

			Number of Nodes			*Time(sec)*			*% Fixed*
$\|H\|$	FC	α	$CPLEX$	BB_{RC}	BB_{PE}	$CPLEX$	BB_{RC}	BB_{PE}	*hubs*
10	L	0.2	0	0	0	0.15	0.04	0.04	100.00
		0.5	0	0	0	0.12	0.06	0.06	100.00
		0.8	0	0	0	0.13	0.06	0.06	100.00
	T	0.2	0	0	0	0.22	0.05	0.05	100.00
		0.5	6	8	0	0.78	0.51	0.44	100.00
		0.8	7	4	0	0.69	0.31	0.30	100.00
20	L	0.2	0	0	0	8.58	2.49	2.49	100.00
		0.5	0	0	0	6.03	1.70	1.70	100.00
		0.8	0	10	0	4.25	5.21	4.04	100.00
	T	0.2	0	0	0	13.51	1.36	1.36	100.00
		0.5	0	0	0	6.85	1.38	1.38	100.00
		0.8	6	2	0	13.64	3.50	3.44	100.00
25	L	0.2	0	0	0	56.46	6.21	6.21	100.00
		0.5	0	14	0	30.50	11.58	7.24	100.00
		0.8	0	18	0	16.65	17.08	9.58	100.00
	T	0.2	0	6	0	119.01	11.92	9.97	100.00
		0.5	0	2	0	53.60	9.15	9.70	100.00
		0.8	0	0	0	25.18	5.40	5.40	100.00
40	L	0.2	0	0	0	1413.40	36.41	36.41	100.00
		0.5	3	10	0	1287.87	79.97	57.87	100.00
		0.8	0	14	0	426.49	125.18	87.92	100.00
	T	0.2	0	0	0	2190.62	46.89	46.89	100.00
		0.5	0	0	0	702.06	21.87	21.87	100.00
		0.8	0	0	0	279.77	21.33	21.33	100.00
50	L	0.2	n.a.	32	2	n.a.	386.03	232.42	97.46
		0.5	n.a.	2	0	n.a.	189.83	193.96	100.00
		0.8	n.a.	12	0	n.a.	407.53	301.24	100.00
	T	0.2	n.a.	0	0	n.a.	75.02	75.02	100.00
		0.5	n.a.	28	2	n.a.	352.01	202.68	98.03
		0.8	n.a.	52	2	n.a.	659.70	187.72	97.65

Table 2: Results of exact algorithms for instances from 10 to 50 nodes and five time periods

cases except two. Finally, the average percentage of reduction time of LR_{RT} relative to LR is 41%.

In order to analyze the efficiency of our exact algorithms we have run a second series of computational experiments using the first set of instances. These results are summarized in Table 2. The first three columns have the same meaning as in Table 1. The next three columns under the heading *Number of Nodes* provide the number of nodes of the branch-and-bound tree required for solving to optimality the problem for: *i)* the CPLEX MIP optimizer using all its machinery (prepossessing phase, cuts, heuristics), *ii)* our standard branch-and-bound algorithm (BB_{RC}) described in Section 6.1 and *iii)* our branch-and-bound algorithm with the partial enumeration phase (BB_{PE}) described in Section 6.2. The next three columns under the heading *Time(sec)* give the CPU time in seconds needed to obtain the optimal so-

lution for CPLEX, BB_{RC} and BB_{PE}, respectively. Finally, the last column *% Red hubs* gives the percentage of hubs that where fixed at the end of the partial enumeration of BB_{PE}.

The results of Table 2 confirm the efficiency of our exact algorithms. Both BB_{RC} and BB_{PE} are able to obtain the optimal solution of each of the 30 instances. In contrast, CPLEX is only able to solve instances with up to 40 nodes. Moreover, the CPU times for solving to optimality the problem are much better with both BB_{RC} and BB_{PE} algorithms than with CPLEX. As can be seen in the *Time(sec)* columns, both BB_{RC} and BB_{PE} beat CPLEX in 28 of the 30 instances and in all instances, respectively, and for the larger instances both algorithms are at least one order of magnitude faster than CPLEX. Furthermore, BB_{PE} seems to outperform BB_{RT} because it is able to obtain better CPU times for all instances, except one. The last column shows that the partial enumeration phase is really effective in fixing the hub nodes. In 27 of the 30 instances it is able to fix all hub nodes, thus no enumeration is required. For the rest of the 3 instances, BB_{PE} only needs to explore two nodes of the tree. On the contrary, BB_{RT} needs a larger amount of branching nodes to prove optimality.

In order to further analyze the efficiency of our exact algorithms, we have run a series of computational experiments using the second set of instances, ranging from 60 to 100 nodes and five time periods. The results are summarized in Table 3. The columns have the same meaning as in the previous table. However, these results now include the percent deviation, CPU time and percent of fixed hubs of LR_{RT}. Therefore, in the last two columns under the heading *% Fixed hubs*, we differentiate between the percentage of fixed hubs in the LR and in the partial enumeration by using LR_{RT} and PE, respectively.

The results of Table 3 further confirm the efficiency of our proposed algorithms. As can be seen we have proven the optimality of the solution obtained with LR_{RT} in six out of the 30 instances. For the remaining instances, the percent deviation is below 2.2%. Once more, the reduction tests proved to be very effective in fixing hub nodes, particularly for the T-type of instances. In the case of the L-type of instances, the percentage of fixed hubs ranges from 0.5% to 98.3% and the average is 32.4%. For the case of the T-type of instances, the percentage of fixed hubs ranges from 72.1% to

			% Dev	*Number of Nodes*		*Time(sec)*			*% Fixed hubs*	
$\|H\|$	*FC*	α	LR_{RT}	BB_{RC}	BB_{PE}	LR_{RT}	BB_{RC}	BB_{PE}	LR_{RT}	*PE*
60	L	0.2	0.00	0	0	253.78	253.78	253.78	96.27	100.00
		0.5	0.00	0	0	142.52	142.52	142.52	98.31	100.00
		0.8	0.00	0	0	145.33	145.33	145.33	98.31	100.00
	T	0.2	0.19	10	0	137.71	188.30	156.14	89.49	100.00
		0.5	0.24	26	2	121.80	251.76	151.66	86.78	98.46
		0.8	0.00	0	0	133.92	133.92	133.92	98.64	100.00
70	L	0.2	0.76	188	0	1219.04	10766.42	1349.01	24.06	100.00
		0.5	0.59	122	2	1151.32	8163.13	1453.79	17.39	95.87
		0.8	0.76	152	0	1355.28	10034.95	1641.76	17.10	100.00
	T	0.2	0.00	0	0	325.40	325.41	325.41	98.55	100.00
		0.5	0.56	6	2	467.65	492.70	509.52	83.48	99.12
		0.8	0.33	14	2	346.17	441.15	388.83	84.64	98.44
75	L	0.2	2.13	176	2	1705.39	23161.25	4847.96	0.54	97.49
		0.5	1.00	90	2	1502.52	9184.80	2005.61	15.14	98.78
		0.8	0.80	80	0	1522.79	8207.66	1823.72	15.41	100.00
	T	0.2	0.68	102	0	504.35	2035.89	548.34	72.97	100.00
		0.5	0.58	80	2	507.20	1333.20	561.99	82.16	98.96
		0.8	0.56	26	0	485.74	666.14	520.56	80.81	100.00
90	L	0.2	1.04	220	2	3949.44	44983.24	7001.63	9.44	96.22
		0.5	0.71	172	2	3427.76	23426.18	4444.81	26.97	97.56
		0.8	0.43	118	0	3208.93	13351.90	3616.97	36.85	100.00
	T	0.2	0.61	250	22	1243.21	8251.10	1985.34	72.13	95.43
		0.5	0.51	56	6	1028.60	2213.48	1297.58	78.88	97.66
		0.8	0.41	60	2	1065.29	2418.86	1193.73	77.08	98.12
100	L	0.2	1.21	312	2	8076.34	106574.40	16168.74	9.90	95.45
		0.5	1.19	272	10	8662.50	115741.93	19220.28	8.28	93.23
		0.8	0.91	192	4	8450.40	68711.20	15857.56	11.72	97.35
	T	0.2	0.85	74	0	1813.93	4054.97	1908.75	83.84	100.00
		0.5	0.36	8	2	1241.91	1359.07	1306.25	91.92	98.54
		0.8	0.00	0	0	961.91	961.95	961.95	100.00	100.00

Table 3: Results of exact algorithms for instances from 60 to 100 nodes and five time periods

100.0% and the average is 85.4%.

Once again, both BB_{RC} and BB_{PE} are able to obtain the optimal solution of all instances. However, BB_{PE} clearly outperforms BB_{RC} in both the required CPU times and number of nodes in the tree. As can be seen in the last column, the partial enumeration phase is able to fix all hub nodes in 14 out of the 30 instances. For the rest of the instances, the percentage of fixed hubs is always above 95%. Columns *Number of Nodes* indicate that BB_{PE} requires to explore much less nodes than BB_{RC}. The maximum number of explored nodes in BB_{PE} is 22, while in BB_{RC} is 312. As for CPU times, BB_{PE} is always much better than BB_{RC} expect in one single instance in which the difference is lees than 100 seconds. For the rest of the instances, most of the times BB_{PE} is one order of magnitude faster than BB_{RC}.

			% Deviation		*Number of Nodes*		*Time(sec)*			*% Fixed hubs*	
$\|H\|$	FC	α	LP	LR_{RT}	$CPLEX$	BB_{PE}	$CPLEX$	LR_{RT}	BB_{PE}	LR_{RT}	PE
10	L	0.2	0.00	0.09	0	0	0.31	0.91	1.31	1.11	100.00
		0.5	0.00	0.00	0	0	0.33	1.21	1.34	55.56	100.00
		0.8	0.00	0.03	0	0	0.38	0.75	0.94	24.44	100.00
	T	0.2	0.00	0.00	0	0	0.32	0.21	0.21	47.78	100.00
		0.5	0.00	0.03	0	0	0.43	0.88	1.05	33.33	100.00
		0.8	0.00	0.00	0	0	0.27	0.50	0.50	75.56	100.00
20	L	0.2	0.00	0.00	0	0	21.67	5.30	5.30	88.42	100.00
		0.5	0.13	0.16	6	0	49.45	9.61	13.15	16.84	100.00
		0.8	0.00	0.00	0	0	8.50	2.54	2.54	91.58	100.00
	T	0.2	0.00	0.00	0	0	23.30	3.64	3.64	84.74	100.00
		0.5	0.00	0.07	5	0	57.74	11.15	12.87	55.79	100.00
		0.8	0.06	0.17	7	2	48.89	7.35	10.74	38.95	98.44
25	L	0.2	0.00	0.10	0	0	79.44	29.25	34.49	37.92	100.00
		0.5	0.00	0.00	0	0	81.58	6.22	6.23	87.92	100.00
		0.8	0.00	0.02	0	0	64.58	23.14	25.06	72.50	100.00
	T	0.2	0.00	0.10	0	0	195.55	25.92	29.37	58.75	100.00
		0.5	0.00	0.24	0	2	208.16	30.26	39.82	32.08	98.86
		0.8	0.06	0.27	4	0	206.61	18.01	23.42	47.50	100.00
40	L	0.2	0.39	0.76	7	2	12587.05	243.81	1069.89	0.00	94.32
		0.5	0.05	0.22	4	2	4822.09	184.90	268.28	30.26	95.89
		0.8	0.00	0.00	0	0	1734.13	84.03	84.03	96.15	100.00
	T	0.2	0.07	0.91	4	2	10728.40	211.28	732.11	18.72	98.34
		0.5	0.05	0.60	3	6	6582.90	217.06	405.90	18.72	96.43
		0.8	0.00	0.45	0	2	1688.05	187.50	254.37	42.05	98.34
50	L	0.2	n.a.	0.16	n.a.	0	n.a.	424.96	562.18	32.65	100.00
		0.5	n.a.	0.41	n.a.	2	n.a.	421.91	850.44	12.04	98.32
		0.8	n.a.	0.54	n.a.	2	n.a.	549.44	1479.83	4.29	97.65
	T	0.2	n.a.	0.09	n.a.	0	n.a.	277.48	311.68	74.90	100.00
		0.5	n.a.	0.42	n.a.	0	n.a.	311.20	378.09	54.49	100.00
		0.8	n.a.	1.00	n.a.	2	n.a.	371.97	645.29	39.80	98.32

Table 4: Results of exact algorithms for instances from 10 to 50 nodes and 10 time periods

In order to further analyze the efficiency and robustness of our best exact algorithm (BB_{PE}), we have run a final series of computational experiments using the third and fourth sets of instances, ranging from 10 to 50 nodes and from 60 to 100 nodes and 10 time periods, respectively. The results are summarized in Tables 4 and 5. The columns have the same meaning as in the previous tables.

The results of Table 4 confirm the efficiency and robustness of our proposed algorithms. As can be seen we have proven the optimality of the solution obtained with LR_{RT} in 8 out of the 30 instances. For the remaining instances, the percent deviation is below 1.0%. Once again, it can be seen that the LP bound of model LM is very tight since it is able to close the duality gap in 17 out of the 24 instances with up to 40 nodes. As expected,

for the 50-node instances, CPLEX is not able to load the problem because of memory requirements. Once more, the reduction tests proved to be effective in fixing hub nodes. In the case of the L-type of instances, the percentage of fixed hubs ranges from 0.0% to 96.1% and the average is 43.4%. For the case of the T-type of instances, the percentage of fixed hubs range from 18.7% to 84.7% and the average is 48.2%.

Once again, BB_{PE} is able to obtain the optimal solution of all instances. In contrast, CPLEX is only able to solve to optimality instances with up to 40 nodes. Furthermore, BB_{PE} is much faster than CPLEX for all instances except in the smaller ones of 10 nodes. As can be seen in the last column, the partial enumeration phase is able to fix all hub nodes in 20 out of the 30 instances. For the rest of the instances, the percentage of fixed hubs is always above 94% and the number of explored nodes is always two except for one instance with six nodes.

The results of Table 5 assess the efficiency of our proposed approach for solving large-size instances. Given the dimension of the considered instances, LR_{RT} is only able to prove the optimality of the obtained solutions in 3 out of the 30 instances and for the rest of the instances the percent deviation never exceeds 2% except for one single instance which has 2.77% of deviation. Once again the reduction tests proved to be effective in fixing hub nodes, particularly for the T-type instances. In the case of the L-type instances, the percentage of fixed hubs ranges from 0.0% to 40.6% and the average is 9.5%. For the case of the T-type instances, the percentage of fixed hubs range from 53.4% to 98.6% and the average is 71.5%.

The BB_{PE} algortihm is again able to obtain the optimal solution of all instances. As can be seen in the last column, the partial enumeration phase can fix all hub nodes in 6 out of the 30 instances and for the rest of them, the percentage of fixed hubs is always above 86.5%. Taking into account the size of the instances, the number of explored nodes is considerably small. In particular, for 28 out of 30 of the instances the number of branching nodes is always less than 24 and for the other two instances is less than 252.

			% Dev	*Nodes*	*Time(sec)*		*% Fixed hubs*	
$\|H\|$	FC	α	LR_{RT}	BB_{PE}	LR_{RT}	BB_{PE}	LR_{RT}	PE
60	L	0.2	0.26	2	1108.97	1523.87	24.58	97.34
		0.5	0.35	12	1263.44	2162.50	14.41	94.56
		0.8	0.20	0	1378.70	1697.07	15.76	100.00
	T	0.2	0.06	0	340.67	386.09	87.46	100.00
		0.5	0.22	2	376.87	654.40	80.51	98.23
		0.8	0.30	2	320.25	502.13	74.41	98.23
70	L	0.2	0.53	2	3607.09	5756.87	8.55	96.43
		0.5	0.35	2	3376.94	5760.98	10.87	95.27
		0.8	0.13	0	2166.26	2495.50	40.58	100.00
	T	0.2	0.00	0	473.97	474.02	96.67	100.00
		0.5	0.00	0	499.28	499.31	98.55	100.00
		0.8	0.00	0	352.05	352.08	98.55	100.00
75	L	0.2	0.92	4	4343.41	19706.27	1.62	95.34
		0.5	1.83	2	4463.57	21365.67	0.00	97.65
		0.8	0.89	2	3907.23	21532.27	3.51	96.55
	T	0.2	1.38	12	1354.38	3511.32	60.54	93.21
		0.5	0.96	2	1486.34	2530.99	59.73	98.34
		0.8	1.30	24	1325.52	2723.43	64.19	95.76
90	L	0.2	1.01	2	8714.84	31231.50	2.81	94.36
		0.5	0.67	2	7903.07	17902.46	11.12	95.43
		0.8	0.84	6	8868.38	29652.33	7.19	94.36
	T	0.2	1.16	2	3015.72	5330.32	56.63	97.89
		0.5	0.82	6	3016.10	5601.41	61.69	96.98
		0.8	1.42	6	3113.38	6701.70	53.37	96.98
100	L	0.2	2.77	252	15437.98	291925.06	0.00	86.48
		0.5	1.71	18	15290.67	225999.74	0.30	96.52
		0.8	1.50	156	15440.98	241895.49	1.31	88.60
	T	0.2	1.78	2	5423.55	14511.74	54.04	97.90
		0.5	1.24	2	4988.45	8323.31	62.12	99.00
		0.8	1.45	2	5154.50	7963.00	63.94	99.10

Table 5: Results of exact algorithms for instances from 60 to 100 nodes and 10 time periods

8. Conclusions

In this paper we have introduced the dynamic uncapacitated hub location problem. The problem was tackled by means of a branch-and-bound algorithm that uses a Lagrangean relaxation as a bounding procedure at the nodes of the enumeration tree. Computational results confirm the efficiency and robustness of the proposed approach. Benchmark instances involving up to 100 nodes and 10 time periods were solved to optimality. To the best of the authors' knowledge, this is the first attempt at solving a discrete dynamic version of the hub location problem.

Acknowledgments

This work was partly founded by the Canadian Natural Sciences and Engineering Research Council under grants 227837-09 and 39682-05. This support is gratefully acknowledged.

References

M. Albareda-Sambola, E. Fernández, Y. Hinojosa, J. Puerto, The multi-period incremental service facility location problem, Computers & Operations Research, 36 1356–1375 (2008).

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

A. Billionnet, S. Elloumi, Placement de tâches dans un système distribué et dualité lagrangienne, RAIRO (Recherche Opérationelle), 26 83–97 (1992).

U. Brännlund, A generalized subgradient method with relaxation step, Mathematical Programming, 71 207–219 (1995).

P.M. Camerini, L. Fratta, F. Maffioli, On improving relaxation methods by modified gradient techniques, Mathematical Programming Study, 3 26–34 (1975).

J.F. Campbell, Locating transportation terminals to serve an expanding demand, Transportation Research Part B. 3 173–192 (1990).

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

J.F. Campbell, A.T. Ernst, M. Krishnamoorthy, Hub location problems, in: Facility Location: Applications and Theory, Z. Drezner, H.W. Hamacher (Eds.) pp. 373–408 Springer, Heidelberg, 2002.

L. Cánovas, M. Landete, A. Marín, New formulations for the uncapacitated multiple allocation hub location problem, European Journal of Operational Research, 172 274–292 (2006).

P. Chardaire, A. Sutter, M.C. Costa, Solving the dynamic facility location problem, Networks, 28 117–124 (1996).

I. Contreras, J.A. Díaz, E. Fernández, Lagrangean relaxation for the capacitated hub location problem, OR Spectrum, 31 483–505 (2009).

H. Crowder, In: Computational improvements for subgradient optimization, Symposia Mathematica, Vol. XIX, Academic Press, London, 1976.

J.R. Current, S.J. Ratick, C.S. ReVelle, Dynamic facility location when the total number of facilities is uncertain: A decision analysis approach, European Journal of Operational Research, 110 597–609 (1998).

Z. Drezner, Dynamic facility location: the progressive p-median problem. Location Science, 3 1–7 (1995).

A. Ernst, M. Krishnamoorthy, Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem, European Journal of Operational Research 104 100–112 (1998).

H.W. Hamacher, M. Labbé, S. Nickel, T. Sonneborn, Adapting polyhedral properties from facility to hub location problems, Discrete Applied Mathematics, 145 104–116 (2004).

A. Marín, Uncapacitated Euclidean hub location: Strengthened formulation, new facets and a relax-and-cut algorithm, Journal of Global Optimization 33 393–422 (2005).

M.T. Melo, S. Nickel, F. Saldanha da Gama, Dynamic multi-commodity capacitated facility location: A mathematical modeling framework for strategic supply chain planning, Computers & Operations Research, 33 181–208 (2006).

M.E. O'Kelly, Hub facility location with fixed costs, Papers in Regional Science, 20 293–306 (1992).

J. Rhys, A selection problem of shared fixed costs and networks, Management Science, 17 200–207 (1970).

H.D. Sherali, O. Ulular, A primal-dual conjugate subgradient algorithm for specially structured linear and convex programming problems, Applied Mathematics and Optimization, 20 193–221 (1989).

D. Skorin-Kapov, J. Skorin-Kapov, M.E. O'Kelly, Tight linear programming relaxations of uncapacitated p-hub median problems, European Journal of Operational Research, 94 582–593 (1997).

T.J. Van Roy, D. Erlenkotter, A dual-based procedure for dynamic facility location, Management Science, 28 1091–1105 (1982).

A. Warszawski, Multi-dimensional location problems, Operational Research Quarterly, 24 165–179 (1973).