Models and Solution Methods for the Uncapacitated r-Allocation p-Hub Equitable Center Problem

Juanjo Peiró
Departament d'Estadística i Investigació Operativa, Universitat de València, Spain

Ángel Corberán
Departament d'Estadística i Investigació Operativa, Universitat de València, Spain

Manuel Laguna
Leeds School of Business, University of Colorado Boulder, USA

Rafael Martí
Departament d'Estadística i Investigació Operativa, Universitat de València, Spain

Abstract

Hub networks are commonly used in telecommunications and logistics to connect origins to destinations in situations where a direct connection between each origin-destination (o-d) pair is impractical or too costly. Hubs serve as switching points to consolidate and route traffic in order to realize economies of scale. The main decisions associated with hub-network problems include 1) determining the number of hubs (p), 2) selecting the p nodes in the network that will serve as hubs, 3) allocating non-hub nodes (terminals) to up to r hubs, and 4) routing the pairwise o-d traffic. Typically, hub location problems include all four decisions while hub allocation problems assume that the value of p is given. In the hub median problem the objective is to minimize total cost while in the hub center problem the objective is to minimize the maximum cost between origin-destination pairs. We study the uncapacitated (i.e., links with unlimited capacity) r-allocation p-hub equitable center problem (with $1 < r < p$) and explore alternative models and solution procedures.

Keywords: facility location, r-allocation p-hub, quality of service, heuristic optimization

Version: July 7, 2016

Corresponding author:
Rafael Martí (rafael.marti@uv.es). *Facultat de Ciències Matemàtiques. Universitat de València. Carrer del Doctor Moliner, 50. 46100 – Burjassot (Valencia), Spain.*

1. Introduction

Hub-and-spoke architectures (aka hub networks) are frequently used in transportation, telecommunications, and computer networks to route traffic between origins and destinations. In hub networks, the number of routes used to connect a network with n nodes is linear with respect to n, resulting in a more efficient use of transportation or communication resources when compared to a point-to-point network for which the number of routes is quadratic with respect to n. Hub networks use transshipment, consolidation, and sorting facilities, generically known as *hubs*, to connect a large number of origin-destination (o-d) pairs and achieve cost efficiencies due to: 1) a reduced number of transportation/communication links, 2) centralized handling and sorting operations, and 3) economies of scale through flow consolidation. Good surveys on this topic are the articles by Alumur and Kara (2008), Campbell and O'Kelly (2012) and Farahani et al. (2013).

There are four main decisions associated with the design of a hub network: 1) determining the value of p, the number of hubs, 2) selecting the p nodes that will be designated as hubs, 3) assigning the non-hub nodes (terminals) to r hubs, 4) routing the traffic through the network to satisfy the o-d pairwise demands. In transportation and telecommunication applications, the traffic may consist of packages, passengers, consumer goods, or electronic data. In hub networks, some direct communication may exist, for instance, when a hub happens to be the destination for a particular demand. In general, however, it is not expected that an o-d demand will reach its destination directly, i.e., without going through at least one hub. Empirical evidence shows that it is neither practical nor economically viable to connect all origins and destinations of a network directly, e.g., airlines do not use direct flights to move all of their passengers or internet providers do not physically connect every pair of computers for data transmission. Instead, depending on the industry, hub networks provide services such as:

- Connecting passengers from multiple origins to multiple destinations through hub airports like Atlanta and Frankfurt.
- Sending express packages via consolidating and sorting facilities (e.g., Memphis for FedEx).
- Enabling data transmission via switches, concentrators, and routers.

The hub network literature is divided into two main areas, location and allocation problems. In general, location problems include all four aforementioned decisions. Allocation problems typically assume that the first decision has been made and therefore operate with a fixed value of p. Location and allocation problems typically assume that hubs are connected by a complete network and that terminals are not connected directly. Another key distinction among hub network design problem relates to the functional form of the objective to be optimized:

- Median: Minimize the total cost
- Center: Minimize the maximum cost between origin-destination pairs
- Covering: Minimize the cost of covering the demand

Campbell (1994) presents several mixed-integer programming formulations for two special cases of these problems, one where $r = 1$ (single allocation problem) and the other where $r = p$ (multiple allocation problem). Campbell recognized the advantage of the multiple allocation model but at the same time realized that there is a cost associated with connecting a terminal to all hubs. He, therefore, proposed mixed-integer programming models with minimum flow thresholds for traffic between terminals and hubs and a fixed cost for using spoke links. Fixed costs are also charged to hubs in the covering models. Ignoring the spoke-link fixed costs, the p-hub median model with flow thresholds can be transformed to the single allocation p-hub median problem setting the flow threshold between a terminal and a hub equal to the terminal demand. Likewise, the model can be transformed to the multiple allocation problem by setting the flow thresholds to zero. Instead of adding fixed costs or flow thresholds, Yaman (2011) introduced models for which $1 < r < p$. In particular, she defined the uncapacitated r-allocation p-hub median problem (UrApHMP) as follows: "Given a set of nodes with pairwise demands, choose p hubs and allocate each node to at most r hubs to minimize the total routing cost."

The following notation is used in the formulation below:

- V is the set of nodes in the network ($|V| = n$), and $H \subseteq V$ is the set of hubs ($|H| = p$).
- t_{ij} is the amount of traffic that needs to be transported from node i to node j.
- c_{ik} is the cost of transporting a unit of traffic from node i to node k. Similarly, c_{kl} and c_{lj}.
- $z_{ik} = 1$ if terminal i is allocated to hub k, and 0 otherwise. By definition, $z_{kk} = 1$ if node k is a hub and $z_{kk} = 0$ otherwise.
- x_{ijkl} is the proportion of the traffic t_{ij} that travels along the path $i \to k \to l \to j$, where nodes k and l are hubs.

The MIP formulation of the UrApHMP in Yaman (2011) is:

$$\text{Min} \quad \textstyle\sum_{ijkl} t_{ij}(\chi c_{ik} + \alpha c_{kl} + \delta c_{lj}) x_{ijkl} \tag{1}$$

$$\text{s.t} \quad \textstyle\sum_k z_{ik} \le r \qquad \forall i \tag{2}$$

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

$$\textstyle\sum_k z_{kk} = p \tag{4}$$

$$\textstyle\sum_k \sum_l x_{ijkl} = 1 \qquad \forall i, j \tag{5}$$

$$\textstyle\sum_l x_{ijkl} \le z_{ik} \qquad \forall i, j, k \tag{6}$$

$$\textstyle\sum_k x_{ijkl} \le z_{jl} \qquad \forall i, j, l \tag{7}$$

$$z_{ik} \in \{0,1\} \qquad \forall i, k \tag{8}$$

$$x_{ijkl} \ge 0 \qquad \forall i, j, k, l \tag{9}$$

In this formulation, constraints (2) ensure that each node is allocated to at most r hubs. Node i can only be allocated to node k if node k is a hub, as imposed in constraint (3). The limit in the number of hubs is established by constraint (4). All traffics must be transported, as stated in constraint (5). Finally, constraints (6) and (7) are associated with the routing of the traffic between each pair of nodes i, j through their corresponding hubs k, l.

As shown in Yaman's literature review, published work has focused on both the single and multiple allocation versions of the p-hub problems. See for example the papers by Brimberg et al. (2015, 2016), Ernst et al. (2009), Hwang and Lee (2012), Ilić et al. (2010), Kara and Tansel (2000), and Milanović (2010). The cost-minimizing solutions found with p-hub median models, however, might include routes for o-d demand pairs that are unreasonably long, calling into question the quality of the service that the network design will provide to some of its customers. The p-hub center problem addresses this issue by minimizing a function of the cost of meeting individual o-d demands. The typical cost structure for a given $i \to k \to l \to j$ path in a hub network is represented in Figure 1, where i is the origin, j is the destination, and k and l are hubs.

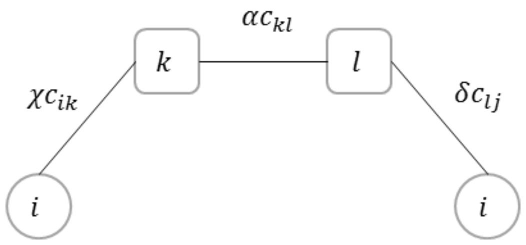

Figure 1 Cost structure of demand from origin i to destination j.

Figure 1 shows the collection cost $c_{ik}$from a terminal i (origin) to a hub k, the transfer cost c_{kl} from hub k to hub l, and the distribution cost c_{lj} from hub l to terminal j (destination). Discount factors, if applicable, are associated with each cost: χ for the origin to hub cost, α for the hub to hub cost, and δ for the hub to destination cost. The total cost of sending one unit of traffic from origin i to destination j via hubs k and l is given by $C_{ijkl} = \chi c_{ik} + \alpha c_{kl} + \delta c_{lj}$. Since x_{ijkl} is the fraction of the traffic t_{ij} from i to j routed via hubs k and l, then the objective function for a basic model of the p-hub median problem has the following form:

$$\min \textstyle\sum_{i,j,k,l} t_{ij} C_{ijkl} x_{ijkl} \tag{10}$$

Note that (10) is a more compact representation of (1) but both objective functions are the same, i.e., they attempt to minimize the total cost. In contrast, in the p-hub center problem, x_{ijkl} is equal to 1 if the demand t_{ij} from origin i to destination j is routed via hubs k and l, and it is equal to 0 otherwise. The objective function for a basic model of the p-hub center problem has the following mathematical form:

$$\min \max_{i,j,k,l} C_{ijkl} x_{ijkl} \tag{11}$$

This objective function is relevant in hub networks where the flow involves time-sensitive commodities, e.g. people, live animals, and perishable items. In this context, cost refers to time, and α may be interpreted as a time discount factor due to higher speed on the inter-hub links (Campbell 1994). An alternative objective function for the p-hub center problem has the following form:

$$\min \max_{i,j,k,l} \left\{ \max\left\{ \chi c_{ik}, \alpha c_{kl}, \delta c_{lj} \right\} x_{ijkl} \right\} \tag{12}$$

In (12) the cost also refers to time and the function is of interest in situations where the maximum travel time in any link is important. For instance, the model would be applicable to networks where items require some special processing that is only available at the hubs.

Minimizing a maximum value is a well-known technique to obtain “balanced” solutions and thus to deal with models related to service. This is the case of the p-hub center problem, which provides a standard approach to overcome the limitations of the p-hub median problem when producing long routes. However, we have empirically found that the p-hub center problem does not model well air transportation problems. The main issue arises with the o-d cities that are away from each other, since a model focusing on minimizing the maximum length route does not optimize medium and lower distance routes. On the other hand, the p-hub median problem minimizes the sum of the costs by selecting hubs which “work well” on average over all the routes. However, we have observed that these hubs may perform poorly on some routes. If the flight company solves the p-hub median problem, it would obtain a solution that is the best in terms of minimizing the overall operational cost, although some of their customers will see that the hub connection proposed in this solution is far from being “ideal”. Hence, it can be expected that these customers consider another flight company with a better service for their specific route. In this paper, we propose a new model based on the p-hub center problem including an equity measure for all o-d pairs.

2. Problem Definition

We are interested in studying a version of the uncapacitated r-allocation p-hub center problem that applies to situations for which an element of equity or fairness is important. In particular, given a network with link costs (distances), our model is based on an estimate of the best route for each origin-destination pair (i,j). It is clear that the customers traveling from i to j desire that the flight company selects i and j as hubs. Since this is no factible in practice for all of them, we consider that the best “realistic” route from i to j is the one with minimum cost in which neither i nor j are hubs. In other words, we consider that the traffic from i to j is routed through the best hubs k and l minimizing the cost of the route $i \rightarrow k \rightarrow l \rightarrow j$, where $k \neq i, l \neq j$. We call this minimum cost the “ideal cost” of pair (i,j). We assume that it is desired to find customer-oriented solutions for which the actual cost for each demand remains relatively close to its ideal minimum cost. This minimum cost, denoted by $\hat{C}_{ij}$, is calculated as follows:

$$\hat{C}_{ij} = \min_{k,l,k\neq i,l\neq j} (\chi C_{ik} + \alpha C_{kl} + \delta C_{lj}) \tag{13}$$

Comparing the cost of a route with an “ideal” cost is a common practice for route planners in order to retain customers. In fact, in domestic flights, planners usually consider alternative transportation modes, such as trains, in their cost analysis. It makes little sense to offer a route much longer than the one offered by a competitor.

Objective function (10) is clearly not equipped to produce customer-oriented solutions since it focuses on minimizing the total cost of the company. Objective functions (11) and (12) may produce some customer-oriented solutions for the largest routes, but ignoring lower cost routes. A customer-oriented solution is defined as one that seeks to minimize large deviations between actual and ideal cost for all demands. Since cost basis may be significantly different from one demand to another, we focus on the following model based on relative deviations:

$$\text{Min} \max_{i,j,i\neq j,t_{ij}>0} \left\{ \frac{\sum_{k,l} c_{ijkl} x_{ijkl} - \hat{c}_{ij}}{\hat{c}_{ij}} \right\} \tag{14}$$

s.t (2) – (8)

$$x_{ijkl} \in \{0,1\} \quad \forall i,j,k,l \tag{15}$$

We refer to this problem as the Uncapacitated r-Allocation p-Hub Equitable Center Problem, which following Yaman (2011), we abbreviate as UrApHECP. This problem arises in competitive environments, such as in the airline industry. As mentioned, an airline interested in minimizing total cost, might design a network for which some itineraries could offer low value to their customers. Value in this context could be interpreted as the relationship between cost and trip duration. That is, a long itinerary (with several stops) with a high cost may be perceived as offering low value. We assume that, all things being equal (e.g., airfare), travelers would typically prefer a short trip duration.

To illustrate this point, consider the network with 55 nodes in the Australian Post (AP) instance introduced by Ernst and Krishnamoorthy (1996). The network and demands correspond to the Australian post services that operate from five hubs (i.e., $p = 5$) and in which terminals may be assigned to no more than two hubs (i.e., $r = 2$). The solution procedures developed by Peiró et al. (2014) and Martí et al. (2015) are designed to minimize the total cost $f(s)$ of a solution s, calculated as given by the objective function (1) of UrApHMP. The best solutions obtained by the application of both procedures include o-d pairs for which their cost is significantly larger than their minimum cost. Table 1 shows five o-d pairs, their associated hubs, as well as the relative difference D between the minimum cost and the solution cost obtained with the procedure of Martí et al.:

$$D_{ij} = 100 \frac{c_{ijkl} - \hat{c}_{ij}}{\hat{c}_{ij}} \tag{16}$$

Origin i	Destination j	Hub k	Hub l	D_{ij}
30	29	36	40	493.9%
30	20	36	40	467.4%
30	19	36	16	449.6%
33	32	40	40	433.3%
33	44	40	40	417.9%

Table 1 Deviation values obtained by solving the UrApHMP.

As the costs are related to distance, the customers associated with the five o-d demands in Table 1 will most likely find their assigned routing unacceptable and might choose to hire a different service. Table 2 shows a different set of assignments for the o-d pairs in Table 1. The relative difference between the actual costs and the minimum costs has decreased across all o-d pairs.

Origin i	Destination j	Hub k	Hub l	D_{ij}
30	29	20	20	114.60%
30	20	20	20	5.20%
30	19	20	20	34.16%
33	32	20	20	240.33%
33	44	20	20	269.30%

Table 2 Smaller deviation values for an alternative solution.

The change of hub assignments from Table 1 to Table 2 causes a 0.82% increase in total cost. Clearly, AP must consider the tradeoff between total cost increase and customer satisfaction and retention. With that in mind, we now develop a method for the UrApHECP.

3. A GRASP for the Uncapacitated $\boldsymbol{r}$-Allocation $\boldsymbol{p}$-Hub Equitable Center Problem

The GRASP methodology was developed in the late 1980s by Feo and Resende (1989) and the acronym was coined in Feo and Resende (1995). Algorithm 1 shows the pseudocode of the GRASP framework that we will use to minimize the objective function $g(s)$ associated with a solution s to the UrApHECP. Such a solution is characterized by the hub selection, the hub assignment, and the traffic routing. Each GRASP iteration consists of constructing a trial solution with a greedy randomized procedure (line 4 in Algorithm 1) and then applying local search to the constructed solution (line 5 in Algorithm 1). The construction phase is iterative, randomized, greedy, and adaptive. This two-phase process is repeated until some stopping condition is satisfied (lines 3 to 9 in Algorithm 1). The best solution (s^*) found over all constructions and local searches is returned as the GRASP solution. We refer the reader to Festa and Resende (2011) for a recent survey of this methodology.

```
GRASP()
1     $s^* \leftarrow \emptyset$
2     $g(s^*) = +\infty$
3     **while** stopping criterion not satisfied **do**
4         $s \leftarrow$ ConstructSolution()
5         $s \leftarrow$ ImproveSolution($s$)
6         **if** $g(s) \leq g(s^*)$ **then**
7             $s^* \leftarrow s$
8         **end if**
9     **end while**
10    **return** $s^*$
```

Algorithm 1 GRASP template.

As indicated in Algorithm 1, the design of GRASP entails the customization of the `ConstructSolution` and the `ImproveSolution` functions. Constructing solutions in the context of the UrApHECP includes three major steps: the selection of the p hubs, the allocation of each terminal to at most r hubs, and the routing of all the traffic. In this article we focus on GRASP designs where the hub selection and the initial hub allocation and routing occur in `ConstructSolution` and where a search for improved hub allocations within the same hub selection occurs in `ImproveSolution`. The next sections describe our efforts to create effective construction and improvement functions within the GRASP framework.

3.1 Construction Methods

Our construction procedures require the ideal (minimum) costs for each o-d pair. These calculations are somewhat onerous but they are performed only once and therefore they can be done offline, before the search procedure is executed. Note that for a given o-d pair, all combinations of hubs must be considered in order to find the minimum cost. As part of this process, we record the number of times $q(h)$ that a node h is selected as the ideal hub for any of the terminals. For instance, if the ideal path for a demand from a terminal i to a terminal j traverses hubs $\hat{k}$ and $\hat{l}$ (i.e., $\hat{C}_{ij} = \chi c_{i\hat{k}} + \alpha c_{\hat{k}\hat{l}} + \delta c_{\hat{l}j}$) then both counts $q(\hat{k})$ and $q(\hat{l})$ are increased by one. If the origin or destination is a hub then only the count for the ideal hub connected to the terminal is incremented. That is, the q count does not include the instances when a node is both a hub and a terminal in the minimum cost. Counting those instances does not add information to the attractiveness of a node to be selected as a hub from the point of view of being used by the terminal nodes in the network.

```
C1()
1    $CL \leftarrow V$
2    $H \leftarrow \emptyset$
     Stage 1: Select $p$ hubs
3    while $|H| < p$ do
4        $q_{max} \leftarrow \max_{h \in CL} q(h)$
5        Randomly select $h^* \in RCL = \{h : h \in CL, q(h) \geq \beta_1 q_{max}\}$
6        $H \leftarrow H \cup \{h^*\}$
7        $CL \leftarrow CL \setminus \{h^*\}$
8    end while
     Stage 2: Assign terminals to $r$ hubs
9    Calculate $q_i(h)$ for all $i \in V \setminus H$ and $h \in H$
10   Let $H_i$ for $i \in V \setminus H$ be the set of $r$ hubs with the largest $q_i(h)$ values
     Stage 3: Route traffic through the network
11   For all $(i,j)$ with $t_{ij} > 0$ let $r_{(i,j)}$ be the path $i \rightarrow k \in H_i \rightarrow l \in H_j \rightarrow j$ that
     minimizes $C_{ijkl}$. Let $\Pi = \{\pi_{(i,j)} \forall (i,j): t_{ij} > 0\}$
12   return $s = \{H, H_i \forall i, \Pi\}$
```

Algorithm 2 Construction procedure `C1`.

Algorithm 2 shows the pseudo-code for `C1`, the first of the two solution construction procedures that we have developed for this application. The hub selection is based on q counts. That is, the greedy function is the maximization of q, which is employed as a measure of the node attractiveness. In the first stage, the procedure calculates q_{max} as the maximum value of q for all the nodes in the candidate list (CL). The candidate list contains all nodes in the graph that have not been chosen as hubs in any of the previous iterations of the construction procedure and therefore $CL = V$ at the beginning of the construction. A restricted candidate list (RCL) is then constructed with those candidate nodes whose q

values are “close to” q_{max}, where proximity is determined by the value of the parameter β_1. The next hub is chosen randomly among those nodes in RCL. The selection steps (lines 3 to 8 in Algorithm 2) are repeated p times to produce a set H of hubs with the specified cardinality.

The second stage of the construction consists of assigning terminals to the chosen set of hubs. Once the set H of hubs has been determined, the ideal costs for all the terminal nodes are recalculated taking into account the hubs defining H. Then, values $q_i(h)$, representing the number of times that hub $h \in H$ appears in the ideal cost for terminal i, when i is either an origin or a destination, are computed. The r hubs with the largest $q_i(h)$ values define the set H_i.

In the third and last stage of the construction the traffic is routed through the network configured by the choices made in the first two stages. Since the links in the network have no capacity limits, a greedy routing is optimal for the chosen hubs and hub allocations. Therefore, each t_{ij} is routed through $k \in H$ and $l \in H$ in order to minimize $C_{ijkl} = \chi c_{ik} + \alpha c_{kl} + \delta c_{lj}$. Finding the set Π of min-cost paths for all o-d pairs requires a computational effort of $O(n^2 r^2)$. The hub selection H, the hub allocation H_i for all $i \in V\backslash H$, and the set of paths Π for all o-d pairs represent a solution s to the problem (see line 12 in Algorithm 2). i.e., $s = \{H, H_i \forall i, \Pi\}$. The objective function value is $g(s) = D_{max} = \max_{(i,j):t_{ij}>0} D_{ij}$. Note that the calculation of D_{ij} as shown in (16) depends on s.

A variant of C1 (referred to as C2) is obtained by modifying the process in which the q values are calculated. The goal of C2, in contrast to C1, is to avoid the selection of inferior hubs. That is, C2 tries to select hubs that are not going to produce large costs. For this purpose, we define $\check{C}_{ij}$ as the largest cost associated with the traffic pair (i,j). The $\check{C}_{ij}$ values are obtained by simply changing $\min$ to $\max$ in expression (13).

Hubs k and l are "acceptable" as hubs for terminals i and j ($t_{ij} > 0$), respectively, if:

$$C_{ijkl} \leq \hat{C}_{ij} + \beta_2(\check{C}_{ij} - \hat{C}_{ij}),$$

where β_2 is set between 0 and 1. In this case, $q(h)$ counts the number of times a node h is selected as "acceptable" for any of the terminals. As in C1, the hub selection step (see line 5 in Algorithm 2) gives preference to candidate nodes with large q values. Note that if $\beta_2 = 0$, C2 coincides with C1, while $\beta_2 = 1$ makes that all p hubs have the same q values (turning the hub selection into a totally random process).

3.2 Local Search Methods

We designed and tested two solution-improvement procedures, LS1 and and LS2. Both of these procedures are local searches, meaning that they stop upon reaching the first local optimal point. Also, both procedures are based on changing the allocation of terminals to hubs, while maintaining the same set of hubs. That is, the hub selection is not changed from the one given by the starting solution generated by one of the construction methods in Section 3.1. LS1 attempts to find an improved solution with respect to objective function (14) by identifying the (i,j) pair for which $D_{ij} = D_{max}$ (see line 4 in Algorithm 3). This is the traffic pair that determines the value of the objective function. Let k and l be the hubs that terminals i and j are currently using to route t_{ij}. Then, a neighborhood search is launched to identify at least one allocation change for i or j that would reduce D_{ij} without increasing the D value for the traffic in o-d pairs $(i,*)$, $(*,i)$, $(*,j)$, and $(j,*)$, where $*$ represents any terminal different from i and j. The complete procedure is shown in Algorithm 3. Line 5 in Algorithm 3 performs the neighborhood search consisting of evaluating different paths for (i,j). If a new path is found, then the relevant sets and values are updated (lines 7 and 8 in Algorithm 3), and the $improve$ flag is set to TRUE. If no new path is found, then the procedure terminates.

```
LS1(s)
1   Compute $D_{ij}$ for all $(i,j)$ pairs such that $t_{ij} > 0$
2   do
3       $improve \leftarrow \mathrm{FALSE}$
4       Identify the path $\pi_{(i,j)}$ for which $D_{ij} = D_{max}$
5       Find a new path $\pi'_{(i,j)}$ through $k' \notin H_i$ or $l' \notin H_j$ such that $D_{i*}, D_{*i}, D_{*j}, D_{j*} <$
        $D_{max}$, where $*$ is any terminal node different from $i$ and $j$.
6       if new path found then
7           Update $H_i$ and/or $H_j$ and $\Pi$
8           Update $D_{i*}, D_{*i}, D_{*j}, D_{j*}$ and $D_{max}$
9           $improve \leftarrow \mathrm{TRUE}$
10      end if
11  while $improve$
12  return $s$
```

Algorithm 3 Solution improvement procedure `LS1`.

The second improvement procedure (`LS2`), summarized in Algorithm 4, does not tackle D_{max} directly as it focuses on minimizing the total cost (1). The neighborhood search consists of evaluating the exchanges $k \in H_i \leftrightarrow k' \notin H_i$ for all i such that $t_{ij} > 0$ or $t_{ji} > 0$, $i,j \in V$, (lines 4 to 13 in Algorithm 4).

```
LS2(s)
1   Compute $D_{ij}$ for all $(i,j)$ pairs such that $t_{ij} > 0$
2   do
3       $improve \leftarrow \mathrm{FALSE}$
4       for $i \in V, k \in H_i, k' \notin H_i$ do
5           Replace $k$ with $k'$
6           Find new paths $\Pi'$ for all $(i,j)$ and $(j,i)$ for which $k \in p_{(i,j)}$ or $k \in p_{(j,i)}$
7           Compute the cost of routing all $(i,j)$ and $(j,i)$ traffic through $\Pi'$
8           if cost has improved then
9               $H_i \leftarrow H_i \setminus \{k\} \cup \{k'\}$
10              Update $\Pi$, $D$ and $D_{max}$
11              $improve \leftarrow \mathrm{TRUE}$
12          end if
13      end for
14  while $improve$
15  return $s$
```

Algorithm 4 Solution improvement procedure `LS2`.

This is done in order to minimize the cost of routing the traffic between i and j (lines 6 and 7 in Algorithm 4). For each exchange, all paths using hub k are recomputed by redirecting the corresponding traffic through the least expensive available route (line 6 in Algorithm 4).

A hub allocation for the terminal under consideration (i.e., node i) is changed to improve the routing cost (line 9 in Algorithm 4). The procedure ends when no change in the hub allocation for any of the terminal nodes is able to reduce the current cost of at least one traffic in the network.

4 The Uncapacitated $\boldsymbol{r}$-Allocation $\boldsymbol{p}$-Hub Median and Equitable Center Problem

Decision makers facing hub network design problems are interested in comparing solutions that trade off cost and customer service. In this context, a bi-objective problem formulation is the appropriate approach to search for solutions that simultaneously consider cost and service. We propose the Uncapacitated r-Allocation p-Hub Median and Equitable Center Problem (UrApHMECP) with the following formulation:

Objective 1	(1)
Objective 2	(14)
Subject to	(2) – (8)
	$x_{ijkl} \in \{0,1\} \quad \forall i, j, k, l$

The goal is to construct the set E of efficient solutions to this bi-objective problem. Efficient solutions are defined in the traditional sense, that is, as those for which it is not possible to decrease the value of one of the objectives without increasing the value of the other objective. (Note that both objectives attempt to minimize a function of the solution.) The outcome of a heuristic approach is an approximation $\hat{E}$ of the efficient frontier E. We propose a `BGRASP` solution method (see Algorithm 5) that builds upon the strategies introduced earlier. The procedure alternates between `C1` and `C2` to construct solutions and then sequentially applies `LS1`, `LS2`, and then again `LS1` (lines 4 to 7 in Algorithm 5). The starting solution for each local search is the best solution found in the previous step. Since `LS1` uses $g(s)$ to measure solution quality and `LS2` uses $f(s)$, the application of both methods is expected to result in a set of solutions that, taken as a whole, produces a reasonable approximation $\hat{E}$ for the UrApHMECP.

```
BGRASP()
1     $\hat{E} \leftarrow \emptyset$
2     while stopping criterion not satisfied do
4         $s \leftarrow$ C1() or C2()
5         $s' \leftarrow$ LS1($s$)
6         $s'' \leftarrow$ LS2($s'$)
7         $s''' \leftarrow$ LS1($s''$)
8         Update $\hat{E}$
9     end while
10    return $\hat{E}$
```

Algorithm 5 Bi-objective GRASP (`BGRASP`).

Although the updating of $\hat{E}$ is shown in line 8 of Algorithm 5, this set of non-dominated solutions is maintained and updated throughout the application of all procedures, that is, after the completion of `C1` or `C2` (which produces s) as well as during all the neighborhood searches associated with `LS1` and `LS2`. Maintaining and updating $\hat{E}$ includes the clean-up processes needed to eliminate solutions that become dominated by the inclusion of new solutions in the set. The second call to `LS1` (see line 7 in Algorithm 5) was added after preliminary experiments showed that additional non-dominated solutions can be found in the trajectory from s'' to s'''.

5 Computational Experiments

This section describes both the scientific and the competitive testing of our proposed solution method. Scientific testing refers to experiments designed to fine-tune the procedure and also to identify the contribution of the various components of the procedure. The goal of these experiments is to determine what makes the procedure perform well (or not) and why. We then engage in the traditional competitive testing to compare performance against alternative methods to search for solutions to the uncapacitated r-allocation p-hub center problem. The procedures in our method have been implemented in C and the integer linear programming formulation described in Section 2 has been solved using CPLEX 12.6.1. All the results reported in this section were obtained by running our codes on an Intel Core i5-4200U PCU @ 1.6 GHz and 8GB of RAM laptop computer with the Ubuntu Linux 14.04.02–64bits operating system. For the single-objective problems, we use the following metrics to measure performance:

- $Value$: Average objective value of the best solutions obtained by the procedure on the instances considered in the experiment.
- Dev: Average percentage deviation from a reference solution, where the reference solution depends on the testing (i.e., scientific or competitive).
- $Best$: Number of instances in a set for which a procedure is able to match the reference solution, where the reference solution depends on the testing (i.e., scientific or competitive).
- CPU: Average computing time in seconds employed by the algorithm.

5.1 Problem Instances

The following two sets of instances have been used:

(1) The **CAB** (Civil Aviation Board) data set. This set is based on airline passenger flows between some large cities in the United States. The 25-node network was introduced by O'Kelly in 1987, who provided distances and passenger flows between nodes. As in other papers, we have used this network to generate 91 instances with $p \in \{3, \ldots, 5\}$, and $r \in \{2, \ldots, p-1\}$. The following discount factors have been widely used: $\chi = 1$, $\delta = 1$, and $\alpha = 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$, and 1.

(2) The **AP** (Australian Post) data set. This set is based on real data from the Australian postal service and was introduced by Ernst and Krishnamoorthy in 1996. The size of the original network is 200 nodes. Smaller instances can be obtained using a code from the ORLIB (Beasley,

1990). As with CAB, many authors have generated various instances from the original network data. We have extended this set of instances by generating 56 more with $n = 30$, 40, 50, 60, 75, 80, 90, and 100 nodes. For these instances, $p \in \{3, \ldots, 7\}$ and $r \in \{2, \ldots, p-1\}$. In line with previous articles, discount factors are set to $\chi = 3$, $\alpha = 0.75$ and $\delta = 2$. These instances have asymmetric flows between the nodes (i.e., for a given pair of nodes i and j, t_{ij} is not necessarily equal to t_{ji}). Moreover, flows from one node to itself can be positive (i.e., t_{ii} can be strictly positive for a given i).

The entire set of instances is available at http://www.optsicom.es, where the instances that were used for scientific testing are identified.

5.2 Scientific Testing

From the complete set of 147 instances corresponding to the CAB and AP data sets described above, we have created a training set of 20 instances of various sizes and values of p and r. Specifically, the training set contains 10 instances from the CAB set, with $n = 20$ $p \in \{3,4,5\}$ and $r \in \{2,3,4\}$, and 10 AP instances with $30 \leq n \leq 90$, $p \in \{5,6,7\}$, and $r \in \{2,3,4,5\}$. The goal of scientific testing is to assess the merit of the various elements included in a solution procedure. Since the tests isolate these elements, it is not expected that the quality of the solutions obtained by these partial procedures will rival those of the best-known (or optimal) solutions that were found with complete search processes. Therefore, for the purpose of scientific testing, it is customary to use as reference solutions in the calculation of Dev the best solutions that the elements being tested are able to produce. This enables the detection of statistical differences between the performance of specific configurations of the elements under study.

In the first experiment, we study the construction methods described in Section 3.1 to assess their merit in terms of solution quality in the context of the UrApHECP. The performance of this solution generators depends on the values of their corresponding parameters, β_1 for `C1` and β_2 for `C2`, which determine the size of the RCL in each construction method. Table 3 shows the Dev values corresponding to `C1` and `C2` for β_1 and β_2 values varying from 0.1 to 1.0.

Each Dev value is calculated over the 20 best solutions constructed by `C1` and `C2`, one for each instance in the training set. The best solution for a problem instance is selected among 100 solutions generated by each method and parameter setting. For example, `C1` with $\beta_1 = 0.1$ is used to generate 100 solutions for the first instance in the training set. The best solution out of these 100 is selected. The process is repeated 19 more times for all instances in the training set. Then, $Dev = 22.9\%$ is the result of averaging the deviation against the reference solution of the 20 best solutions found (one for each problem instance).

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
β_1	22.9%	27.9%	18.9%	20.0%	33.1%	42.9%	61.7%	74.3%	87.7%	94.9%
β_2	17.7%	28.4%	26.2%	32.6%	33.2%	33.1%	32.6%	17.8%	22.6%	21.5%

Table 3 Dev values for `C1` and `C2` solutions of the training set instances

With the goal of detecting differences among the various parameter values, we applied the non-parametric Friedman test for multiple correlated samples. The test was performed on the set of best solutions obtained by each construction method and each parameter value in Table 3. This test computes, for each instance, the rank of each method according to solution quality. Then, it calculates the average rank for each method across all instances. If the averages differ greatly, the associated p-value or level of significance is small. For C1, the Friedman test results in a p-value of 0.000, indicating that there are statistically significant differences among the average solution quality obtained by employing different β_1 values. We then performed a paired sample test (Wilcoxon) for the best solutions obtained by C1 with $\beta_1 = 0.3$ and $\beta_1 = 0.4$, which are the settings that result in the smallest Dev values. The resulting p-value of 0.53 indicates that there is no significant difference between setting β_1 to 0.3 or 0.4. We chose $\beta_1 = 0.3$.

The Friedman test for the best solutions obtained by C2 with the 10 settings of β_2 shown in Table 3 yielded a p-value of 0.13. This result does not provide evidence that there is a significant difference in performance of C2 among all the β_2 values that were tested. Although not supported from a statistical point of view, the values $\beta_2 = 0.1$ and $\beta_2 = 0.8$ resulted in the smallest deviations, and therefore we decided to use $\beta_2 = 0.1$.

We now search for the most effective combination of construction and improvement within GRASP in Algorithm 1. To that end, we test all combinations of construction and improvement procedures, i.e., C1 ($\beta_1 = 0.3$) with LS1 and LS2 as well as C2 ($\beta_2 = 0.1$) with LS1 and LS2. The stopping criterion is set to 100 GRASP iterations for each instance in our training set. Table 4 shows the performance measures associated with each combination.

Combination	$Value$	Dev	$Best$	CPU
C1 ($\beta_1 = 0.3$) + LS1	183.0	6.79%	10	0.8
C1 ($\beta_1 = 0.3$) + LS2	183.6	8.62%	8	13.7
C2 ($\beta_2 = 0.1$) + LS1	186.6	11.19%	11	3.7
C2 ($\beta_2 = 0.1$) + LS2	190.7	15.41%	8	16.6

Table 4 Performance of various GRASP (Algorithm 1) configurations.

The Dev values in Table 4 may be compared with the Dev values in Table 3 for the corresponding β_1 and β_2 settings to measure the contribution of the local search. On average, LS1 and LS2 improve the initial C1 solutions by 12.11% (18.9% - 6.79%) and 10.28% (18.9% - 8.62%), respectively. Likewise, LS1 and LS2 improve the initial C2 solutions by 6.51% (17.7% - 11.19%) and 2.29% (17.7% - 15.41%) respectively. The results in Table 4 indicate that the most effective combination consists of pairing C1 with LS1.

We are also interested in identifying the contribution of the local search procedures within BGRASP. Of particular interest is the marginal improvement obtained by the order in which the local search procedures are applied (as shown in Algorithm 5). Using a termination criterion of 100 iterations, for each instance in the training set we calculate $\hat{E}$ after the construction step (line 4 in Algorithm 5), then after the completion of LS1 and LS2 (line 6 in Algorithm 5), and finally after the completion of the additional call to LS1 (line 7 in Algorithm 5). Figure 2 illustrates the outcome of this experiment on one of the training instances.

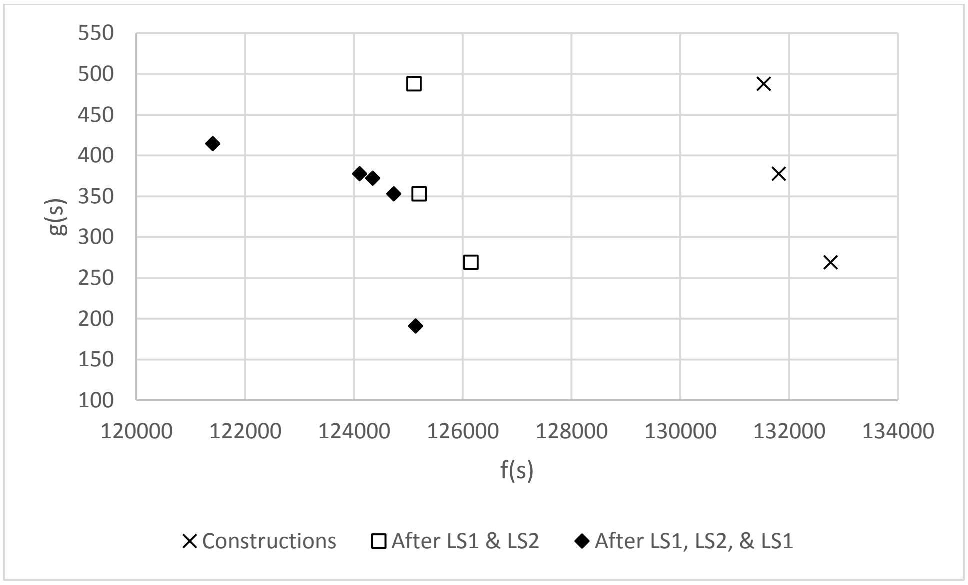

Figure 2 Efficient frontier approximations for an instance in the training set.

Figure 2 shows that each of the local search stages contribute to finding a better approximation $\hat{E}$ (by moving the points toward the left and the bottom of the graph). While this depiction of the local search contributions is only for one of the problem instances in the training set, we have observed a similar behavior in the remaining instances in the set.

5.3 Competitive Testing

As mentioned in Section 2, Martí et al. (2015) developed a solution procedure, based on the scatter search methodology, for the UrApHMP. We argued, through an illustrative example, that a solution procedure designed for the UrApHMP is not capable of producing high quality solutions for the UrApHECP. We now provide experimental evidence to support this argument. The experiment consists of applying Martí et al.'s procedure (SS) and our GRASP (Algorithm 1 with C1 and LS1) to all problems in our test set. The stopping criterion, for both procedures, is set to a maximum number of 1000 iterations for each problem instance. Table 5 shows the average objective function value (i.e., $Value$) and the Dev values calculated against the best-known solutions as the reference points. The set of instances has been divided in six subsets according to the number of nodes.

Set	n	Instances	UrApHMP		UrApHECP		CPU Seconds	
			$Value$ (SS)	Dev (GRASP)	$Value$ (GRASP)	Dev (SS)	SS	GRASP
CAB	15	27	15059745.9	7.6%	48.4	84.0%	0.1	0.6
CAB	25	54	6825596063.6	14.2%	128.1	179.7%	0.9	13.5
AP	40	6	138422.4	6.2%	121.5	37.8%	2.8	46.8
AP	60	10	132012.9	15.3%	271.5	28.7%	11.2	305.3
AP	80	15	129413.6	18.6%	392.1	141.4%	49.9	837.5
AP	100	15	130146.4	23.4%	674.3	56.6%	102.3	2757.5
Total/Average		127	2905471217.6	14.1%	217.8	121.7%	19.4	660.2

Table 5 Performance comparison between SS and GRASP on UrApHMP and UrApHECP.

The procedures perform as expected. SS finds the best-known solutions for all instances when solving the UrApHMP. That is, a Dev column for SS under UrApHMP would consists of all zeros. The third column of Table 5 shows the average objective values of the best-known solutions for UrApHMP. The GRASP solutions to these problems are, on average, 14.1% away from the best-known solutions, as indicated by the values in the Dev (GRASP) column in Table 5. When solving the UrApHECP on the same instances, the performance of SS and GRASP is exactly the opposite to the previous case. That is, GRASP finds all the best-known solutions and the SS solutions are, on average, 121.7% away from these reference points. These results provide empirical evidence that UrApHMP and UrApHECP are significantly different problems, requiring specialized procedures and hence justifying our current work. We point out that the computational effort of GRASP is about an order of magnitude larger than the effort employed by SS. This is due to the complexity of the calculations associated with the optimization of the UrApHECP, which we illustrated in Section 2.

In our next competitive experiment, we attempted to obtain solutions for the UrApHECP by solving the MIP formulation in Section 2. The well-known commercial MIP solver CPLEX 12.6.1 was unable to provide any feasible integer solutions to problems with $n > 20$. Therefore, we eliminated the possibility of comparing the performance of our method against this optimization package. We then attempted a comparison of our GRASP for the UrApHECP with two general-purpose metaheuristic optimizers, LocalSolver and OptQuest[1]. The LocalSolver model that we used for this competitive testing is in the Appendix. The OptQuest code is available from the authors upon request. As a preliminary experiment, both of these optimizers were tried on a small AP instance with $n = 20$, $p = 4$, $r = 2$, $\chi = 3$, $\alpha = 0.75$ and $\delta = 2$. The best-known solutions for UrApHMP and UrApHECP have objective function values of $f(s) = 132263$ and $g(s) = 50.65\%$, respectively. These solutions were found in a fraction of a CPU second by SS (for UrApHMP) and GRASP (for UrApHECP). LocalSolver was able to match these solutions within 20-minute runs. However, OptQuest was only able to find a solution for UrApHMP with an objective function value of $f(s) = 136704.72$ and a solution for UrApHECP with an objective function value of $g(s) = 84.44\%$. Similar results were found with additional small problems and therefore we decided to continue our competitive testing only with LocalSolver.

Given the general nature of LocalSolver, we allowed it to run for 20 minutes for each problem instance, which represents about twice the computational effort of our GRASP. LocalSolver ran into memory issues on problems with $n > 60$, therefore our experiments were limited to the 35 instances summarized in Table 6.

Set	n	Instances	LocalSolver			GRASP		
			$Value$	Dev	$Best$	$Value$	Dev	$Best$
CAB	15	9	60.8	0.0%	9	61.1	0.58%	8
CAB	25	10	235.5	14.4%	2	207.2	0.56%	9
AP	40	6	132.3	15.5%	1	113.5	0.31%	5
AP	60	10	402.5	47.7%	0	271.5	0.00%	10
Total/Average		35	220.6	20.4%	12	172.0	0.36%	32

Table 6 Comparison between LocalSolver and GRASP on the UrApHECP

[1] LocalSolver is a product of Innovation 24 (localsolver.com) and OptQuest is a product of OptTek Systems (opttek.com).

The quality of the solutions that LocalSolver finds decreases with the size of the problem. For the CAB instances, LocalSolver's performance is somewhat comparable to GRASP. However, the gap widens when tackling the larger AP instances.

To the best of our knowledge, the literature does not include a procedure specifically developed for the UrApHMECP. Once again, we started by executing both LocalSolver and OptQuest on a small AP problem with $n = 20$. The narrow multiobjective capabilities of LocalSolver limit our analysis. LocalSolver is not designed to search for an approximation of an efficient frontier. If more than one objective is defined within a LocalSolver model, the system treats them lexicographically. The objectives are considered in the order that they are declared. The execution of LocalSolver produces a single solution that is the best approximation of the lexicographic optimization of the objectives. For each problem instance, we attempt to produce four non-dominated solutions in the efficient frontier: 1) minimize $f(s)$ (modelNumber = 1 in the Appendix), 2) minimize $g(s)$ (modelNumber = 2), 3) minimize $g(s)$ and then $f(s)$ (modelNumber = 3), and 4) minimize $f(s)$ and then $g(s)$ (modelNumber = 4). Given the heuristic nature of the LocalSolver, there is no guarantee that the four solutions will be non-dominated and therefore we remove those that are dominated.

OptQuest has a multiobjective setting in which the procedure searches for an approximation of the efficient frontier associated with the objective functions defined in the optimization model. Figure 3 shows the non-dominated solutions found by BGRASP (5), LocalSolver (2), and OptQuest (1) on an AP problem with $n = 20$.

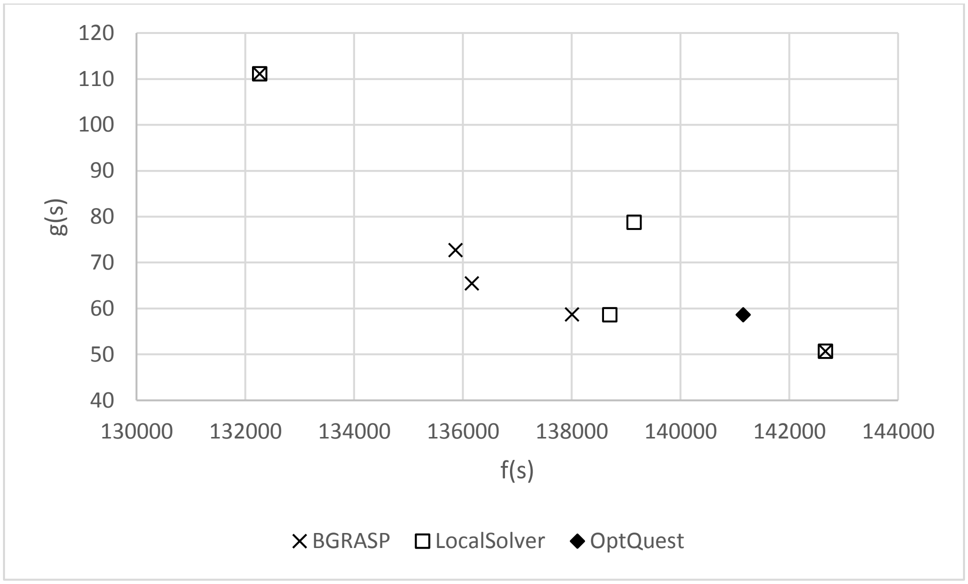

Figure 3. Bi-objective solutions to an AP instance with $n = 20$.

The BGRASP solutions in Figure 3 dominate two of the LocalSolver solutions, the ones found using the lexicographical multiobjective functionality of LocalSolver. The solution found using the multiobjective OptQuest search is also dominated by the GRASP solutions. In addition, one of the LocalSolver solutions dominates the OptQuest solution. Since similar results were found with other smaller instances, we focused our competitive testing on LocalSolver.

We employ the same 35 instances in Table 6 to run BGRASP for 1000 iterations and LocalSolver for 20 minutes with modelNumber = 1, 3, and 4. The solutions for modelNumber = 2 were obtained from the experiments reported in Table 6. The comparison is done in terms of the hypervolume. This metric was developed by Zitzler and Thiele (1999) and measures the size of the space covered, which approximates the volume where the dominated points reside. Hence, the larger the hypervolume the better. The number of points in $\hat{E}$ is another measure of interest in multiobjective optimization. Table 7 reports both the average hypervolume and the average number of points found by BGRASP and LocalSolver for each subset of problems.

Set	n	Instances	Hypervolume		Number of Points	
			BGRASP	LocalSolver	BGRASP	LocalSolver
CAB	15	9	0.29	0.11	1.89	1.78
CAB	25	10	0.84	0.15	6.70	2.50
AP	40	6	0.77	0.03	3.67	1.83
AP	60	10	0.71	0.20	4.40	1.80
Total/Average		35	0.65	0.13	4.29	2.00

Table 7 Comparison between LocalSolver and BGRASP on the UrApHMECP

The nondominated solutions that BGRASP finds result in a hypervolume that on average is about 5 times larger than the hypervolume corresponding to the LocalSolver solutions. Also on average, half of the solutions that LocaSolver finds for each problem instance are dominated (as indicated by the value of 2.00 in the last row of the LocalSolver column under the Number of Points heading).

6 Conclusions

We tackle two hub-network design problems that have not been addressed in the literature, the Uncapacitated r-Allocation p-Hub Equitable Center Problem (UrApHECP) and the Uncapacitated r-Allocation p-Hub Median and Equitable Center Problem (UrApHMECP). Modeling equity as a relative deviation from an ideal value (e.g., cost) is applicable in contexts where solutions for which some of the demand is fulfilled using routes that are far from ideal are not desirable. We argue that the airline industry is such that if routes connecting two terminal nodes (i.e., an origin and a destination) are far from ideal (e.g., a direct flight) it could result in a loss of customers to the competition. In order to keep operational costs in perspective, we suggest to formulate the problem as a bi-objective optimization model that accounts for both cost-efficiency and service.

The UrApHECP is a so-called minimax model, since it seeks to minimize the maximum deviation from the ideal values. These problems create "flat" objective function spaces because many solutions share the same objective function value. Empirical evidence points to multistart methods as an effective way of searching these spaces. This is the reason the selection of GRASP as the underlying methodology for our solution procedure. Careful scientific testing was performed to identify a high-performing configuration of our search method. This was followed by a competitive testing designed to show the need for a specialized procedure for both the UrApHECP and the UrApHMECP. Although our tests are limited to problem sizes that LocalSolver is able to handle, the proposed GRASP and BGRASP are scalable and able to tackle problems of realistic size.

Acknowledgments

This work was supported by the Spanish Ministerio de Economía y Competitividad (TIN2015-65460-C02-01, MTM2015-68097, and Predoctoral Grant BES-2013-064245) and by the Generalitat Valenciana (Project Prometeo 2013/049). This support is gratefully acknowledged. Authors also want to thank Julián Molina and José Manuel Colmenar for their help in the computation of the hypervolume measures given in Table 7.

References

Alumur, S. and B.Y. Kara. 2008. Network hub location problems: the state of the art. European Journal of Operational Research 190, no.1: 1-21.

Beasley, J.E. 1990. OR-library: distributing test problems by electronic mail. Journal of the Operational Research Society 41, no. 11: 1069-1072.

Brimberg, J., N. Mladenovic, R. Todosijevic, and D. Urosevic. 2015. A basic variable neighborhood search heuristic for the uncapacitated multiple allocation p-hub center problem. Optimization Letters. In press, http://dx.doi.org/10.1007/s11590-015-0973-5

Brimberg, J., N. Mladenovic, R. Todosijevic, and D. Urosevic. 2016. General variable neighborhood search for the uncapacitated single allocation p-hub center problem. Optimization Letters. In press, http://dx.doi.org/10.1007/s11590-016-1004-x

Campbell, J. F. 1994. Integer programming formulations of discrete hub location problems. European Journal of Operational Research 72, no. 2: 387-405.

Campbell, J.F. and M.E. O'Kelly. 2012. Twenty-five years of hub location research. Transportation Science 46, no. 2: 153-169.

Ernst, A. T., H. Hamacher, H. Jiang, M. Krishnamoorthy, and G. Woeginger. 2009. Uncapacitated single and multiple allocation p-hub center problems. Computers & Operations Research, 36, no. 7: 2230-2241.

Ernst, A. and M. Krishnamoorthy. 1996. Efficient algorithms for the uncapacitated single allocation p-hub median problem. Location Science 4, no. 3: 139–154.

Farahani, R.Z., M. Hekmatfar, A.B. Arabani and E. Nikbakhsh. 2013. Hub location problems: a review of models, classification, solution techniques, and applications. Computers & Industrial Engineering 64, no: 4: 1096-1109.

Feo, T. A. and M. G. C. Resende. 1989. A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters 8, no. 2: 67-71.

Feo, T. A. and M. G. C. Resende. 1995. Greedy randomized adaptive search procedures. Journal of Global Optimization 6, no. 2: 109-133.

Festa, P. and M. G. C. Resende. 2011. GRASP: Basic components and enhancements. Telecommunication Systems 46, no. 3: 253-271.

Hwang, Y. H. and Y. H. Lee. 2012. Uncapacitated single allocation p-hub maximal covering problem. Computers and Industrial Engineering 63, no. 2: 382-389.

Ilić, A., D. Urošević, J Brimberg and N. Mladenović. 2010. A general variable neighborhood search for solving the uncapacitated single allocation p-hub median problem. European Journal of Operational Research 206, no. 2: 289-300.

Kara, B. Y. and B. Ç. Tansel. 2000. On the single-assignment p-hub center problem. European Journal of Operational Research 125, no. 3: 648-655.

Martí, R., Á. Corberán and J. Peiró 2015. Scatter Search for an uncapacitated p-hub median problem. Computers & Operations Research 58, 53-66.

Milanović, M. 2010. A new evolutionary based approach for solving the uncapacitated multiple allocation p-hub median problem. X.Z. Gao (Ed.), et al., Soft Computing in Industrial Applications, AISC 75, Springer-Verlag, Berlin, 81-88.

Peiró, J., Á. Corberán and R. Martí. 2014. GRASP for the uncapacitated r-allocation p-hub median problem. Computers & Operations Research 43, 50-60.

Yaman, H. 2011. Allocation strategies in hub networks. European Journal of Operational Research 211, no. 3: 442-451.

Zitzler, E. and L. Thiele .1999. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation 3, no 4: 257–271.

Appendix

This is the model() function within the LSP file that we used with LocalSolver 6.0 to find solutions to the UrApHMP (modelNumber = 1), UrApHECP (modelNumber = 2), and UrApHMECP (modelNumber = 3 or 4). We do not include our input() and output() function but the entire LSP file is available upon request.

```
function model()
{
  // z[i][k] equal 1 if node i is assigned to hub k
  z[1..n][1..n] <- bool();

  // allocate each node to at least one hub but no more than r hubs
  for [i in 1..n] constraint sum[k in 1..n](z[i][k]) >= 1;
  for [i in 1..n] constraint sum[k in 1..n](z[i][k]) <= r;

  // allocate node i to k only if k is a hub
  for [i in 1..n][k in 1..n] constraint z[i][k] <= z[k][k];

  // select p hubs
  constraint sum[k in 1..n] (z[k][k]) == p;

  // calculate sum of routing costs
  csum <- 0;
  for [i in 1..n][j in 1..n : t[i][j] > 0] {
    cost[i][j] <- min[k in 1..n][l in 1..n](z[i][k]*z[j][l] > 0 ?
    xi*c[i][k] + alpha*c[k][l] + delta*c[l][j] : 99999999);
    csum <- csum + t[i][j]*cost[i][j];
  }

  // identify maximum difference from ideal
  M <- max[i in 1..n][j in 1..n : i != j && t[i][j] > 0]
  ((cost[i][j] - d[i][j]) / d[i][j]);

  // objective function
  if (modelNumber == 1) minimize csum;
  if (modelNumber == 2) minimize M;
  if (modelNumber == 3) {
    minimize M;
    minimize csum;
  }
  if (modelNumber == 4) {
    minimize csum;
    minimize M;
  }
}
```