p-facility Huff location problem on networks[☆]

Rafael Blanquero[a,*], Emilio Carrizosa[a], Boglárka G.-Tóth[b], Amaya Nogales-Gómez[c,1]

[a] *Departamento de Estadística e Investigación Operativa, Facultad de Matemáticas, Universidad de Sevilla, {rblanquero,ecarrizosa}@us.es*
[b] *Budapest University of Technology and Economics, Hungary, bog@math.bme.hu*
[c] *Mathematical and Algorithmic Sciences Lab, Huawei France R&D, Paris, France, amaya.nogales.gomez@huawei.com*

Abstract

The p-facility Huff location problem aims at locating facilities on a competitive environment so as to maximize the market share. While it has been deeply studied in the field of continuous location, in this paper we study the p-facility Huff location problem on networks formulated as a Mixed Integer Nonlinear Programming problem that can be solved by a branch-and-bound algorithm. We propose two approaches for the initialization and division of subproblems, the first one based on the straightforward idea of enumerating every possible combination of p edges of the network as possible locations, and the second one defining sophisticated data structures that exploit the structure of the combinatorial and continuous part of the problem. Bounding rules are designed using DC (difference of convex) and Interval Analysis tools.

In our computational study we compare the two approaches on a battery of 21 networks and show that both of them can handle problems for $p \leq 4$ in reasonable computing time.

Keywords: Huff location problem, location on networks, p-facility, difference of convex, global optimization

[☆]This work has been partially supported by projects MTM2015-65915-R of Ministerio de Economía y Competitividad, Spain, P11-FQM-7603 and FQM-329 of Junta de Andalucía, Spain.

[*]Corresponding author.

[1]Most of this work was done when the author was at Departamento de Estadística e Investigación Operativa, Facultad de Matemáticas, Universidad de Sevilla.

1. Introduction

Competitive location models (Eiselt et al., 1993; Plastria, 2001) were originally introduced by Hotelling (1929), considering the location of two competing facilities on a linear market. In the seminal work of Hotelling, users patronize the facility closest to them. In contrast with this all·or·nothing assumption, it was introduced the Huff location model (Huff, 1964), in which the probability that a user patronizes a facility is proportional to its attractiveness and inversely proportional to a power of the distance to it. The Huff location problem has been extensively studied in the field of continuous location (Blanquero and Carrizosa, 2009; Drezner and Drezner, 2004; Fernández et al., 2007; Huff, 1964, 1966) and successfully applied in the marketing field, in problems such as location of petrol stations, shopping centers or restaurants (Ghosh et al., 1995; Okabe and Kitamura, 1997; Okunuki and Okabe, 2002). The natural extension of this problem to that of locating p-facilities on the plane, has also received certain attention in the literature (Drezner, 1998; Drezner et al., 2002; Redondo et al., 2009a,b; Tóth et al., 2009).

Network optimization models (Bertsekas, 1998) are widely used in practice due to their methodological aspects and intuitive formulations. They arise naturally in the context of assignment, flow, transportation or location problems among others. For a comprehensive introduction to location models on networks see Labbé et al. (1995).

The combination of the Huff location problem and network optimization has been already addressed in the literature (Berman et al., 2011; Blanquero et al., 2014) and applied to market area analysis (Okabe and Kitamura, 1997) and demand estimation (Okabe and Okunuki, 2001). The single-facility case has been solved in Berman et al. (2011) by means of Interval Analysis (IA) bounds, and in Blanquero et al. (2014) using IA and difference of convex (DC) bounds. Different metaheuristics have been proposed for the p-facility case in Roksandić et al. (2012), but no attempt has been made so far to address the multifacility case with exact methods. This lack of progress in the state of the art is due to the difficulty of the problem, caused by its combinatorial component added to the continuous global optimization: one has to decide which edges are to contain facilities, and, for the choice of edges given, the location is to be decided. Thanks to the recent results described in Blanquero et al. (2015), in which a new data structure is introduced to

address multifacility location problems on networks via branch and bound algorithms, we solve in this paper the p-facility Huff location problem on networks, formulated as a Mixed Integer Nonlinear Programming (MINLP) problem.

The remainder of this paper is organized as follows. In Section 2 we set up the notation for networks and introduce the p-facility Huff location problem. In Section 3, a branch-and-bound method with different initialization and branching rules is described. Section 4 is devoted to procedures for calculating lower and upper bounds. Computational results are reported in Section 5, where the p-facility Huff location problem is solved using the different branching and bounding rules for 12 real-life and 9 artificial networks. Finally, Section 6 contains a brief summary, final conclusions and some lines for future research.

2. The model

Let $N = (V, E)$ be a network, with node set V and edge set E. The length of the edge $e \in E$ is denoted by l_e. The distance between two nodes $a_i, a_j \in V$ is calculated as the length of the shortest path (Labbé et al., 1995) from a_i to a_j. For each $e \in E$, with end-nodes a_i, a_j, we identify each $x \in [0, l_e]$ with the point in the edge e at distance x from a_i and $l_e - x$ from a_j. This way, we obtain that, for any vertex $a_k \in V$ and $x \in e$, the distance $d(x, a_k)$ from x to a_k, as a function of x, is a concave piecewise linear function, given by $d(x,\ a_k) = \min\{x + d(a_i,\ a_k),\ (l_e - x) + d(a_j,\ a_k)\}$.

In the p-facility Huff location model, the finite set V of vertices of the network represents users, asking for a certain service. Each user $a \in V$ has demand $\omega_a \geq 0$, that is patronized by different existing facilities, located at points $y_1, \ldots, y_r$ on the network. The demand captured by facility at y_i from user a is assumed to be inversely proportional to a positive nondecreasing function of the distance $d(a, y_i)$, namely, $\alpha_{ai}/(d(a, y_i))^2$ is used as the utility or attraction function of y_i, where $\alpha_{ai} > 0$ denotes the attraction that user a feels towards the facility at y_i. Therefore, the demand captured by the facility at y_i from the user at a is given by

$$\omega_a \frac{\alpha_{ai}/(d(a, y_i))^2}{\sum_{j=1}^{r} \alpha_{aj}/(d(a, y_j))^2}. \tag{1}$$

A new firm is entering the market, by locating p new facilities at some

points $x_1, \ldots, x_p$ on the network. For simplicity, all new facilities are assumed to have the same attractiveness $\alpha_a > 0$, which is fixed. The new facilities perturb how the market is shared, since the new facilities will capture part of the demand from $a \in V$,

$$\omega_a \frac{\sum_{j=1}^{p} \alpha_a/(d(a,x_j))^2}{\sum_{j=1}^{p} \alpha_a/(d(a,x_j))^2 + \sum_{j=1}^{r} \alpha_{aj}/(d(a,y_j))^2}. \tag{2}$$

Our goal is the maximization of the market share of the entering firm. Thus, the problem we need to solve can be formulated as

$$\max_{\substack{x_1 \in [0,l_{e_1}],\ldots,x_p \in [0,l_{e_p}] \\ e_1,\ldots,e_p \in E}} \sum_{a \in V} \omega_a \frac{\sum_{j=1}^{p} \alpha_a/(d(a,x_j))^2}{\sum_{j=1}^{p} \alpha_a/(d(a,x_j))^2 + \sum_{j=1}^{r} \alpha_{aj}/(d(a,y_j))^2}. \tag{3}$$

In order to simplify the previous expression, the following positive constant is considered for each $a \in V$:

$$\beta_a = \sum_{j=1}^{r} \frac{\alpha_{aj}/\alpha_a}{(d(a,y_j))^2}. \tag{4}$$

Problem (3) can be rewritten then as the following MINLP:

$$\max_{\substack{x_1 \in [0,l_{e_1}],\ldots,x_p \in [0,l_{e_p}] \\ e_1,\ldots,e_p \in E}} F(x_1, \ldots, x_p) \tag{5}$$

where F is defined as

$$F(x_1, \ldots, x_p) = \sum_{a \in V} \omega_a \frac{1}{1 + \frac{\beta_a}{\sum_{j=1}^{p} \frac{1}{(d(a,x_j))^2}}}. \tag{6}$$

The MINLP problem (5) is formed by a combinatorial and a continuous part. First, we need to solve the combinatorial problem of choosing a set of p edges to locate the facilities, and then solve a continuous location problem on the edges.

3. The methodology

The natural way to solve the MINLP formulation of the p-facility Huff location problem is to use a branch-and-bound method. We differenciate two main phases: the initialization phase and the branch-and-bound phase. In the initialization phase the initial exploration tree is prepared. In the branch-and- bound phase, an element of the list is selected iteratively (until the termination rule is fulfilled) according to a selection criterion, and then is divided into new elements that are included into the list if they cannot be eliminated by their bounds. In this phase, division, bounding, selection, elimination and termination rules are required.

In this paper we propose different approaches for the initialization phase, division and bounding rules. As selection, elimination and termination rules, we always apply the usual ones from the literature (Berman et al., 2011): the element to be evaluated is selected as the one with the largest upper bound, elements whose upper bound are lower than the current lower bound are eliminated, and the optimization is terminated when the relative error between the largest upper bound and the current lower bound is less than a fixed tolerance. This section is aimed at describing two types of initialization and division rules. Bounding rules will be discussed in Section 4.

The methodology proposed in Blanquero et al. (2014), where the single-facility problem is tackled, has in common with the methodology herein considered the use of a branch-and-bound algorithm with, essentially, the same upper bounds. However, the difficulty introduced by the combinatorial part of the problem leads us to use sophisticated data structures in the branch and bound recently introduced in Blanquero et al. (2015), so that the election of the p edges can be done in an optimal way during the algorithm running. The initialization and the division phases of the algorithm are deeply affected by the use of these structures.

3.1. Total enumeration

The straightforward way of solving Problem (5) is to separate the combinatorial and the continuous part of the problem: we first fix a set of p edges to locate the facilities, and then solve a continuous location problem on the edges. This means the branch-and-bound approach starts with a partition of the search space formed by the cartesian product of p-uples. The p-uples are formed by every possible combination of p edges, taking into account that several facilities can be located at the same edge, i.e., repetitions of

the same edge are allowed in the elements of the partition. But obviously, permutations of the p-uples are not taken into account.

During the algorithm running, the edges forming the initial elements of the partition are going to be divided into small pieces (segments of edge), which will be referred to as *subedges* throughout the paper.

We denote by $\underline{s} = (s_1, \ldots, s_k)$ an element of the partition, where each component s_i is a (sub)edge that has a multiplicity $m(s_i)$, i.e., the number of facilities located at s_i is $m(s_i)$. Hence, $m(s_1) + \ldots + m(s_k) = p$. To avoid symmetric sets, for any element of the partition $\underline{s} = (s_1, \ldots, s_k)$, and any $s_i = [l, u] \subseteq [0, l_e], e \in E, s_i \in \underline{s}$, the cartesian product $\Pi_{j=1}^{m(s_i)} s_i$ is replaced by $\{l \leq x_1 \leq \ldots \leq x_{m(s_i)} \leq u\}$. For example, let us consider $p = 2$ and the edge $e = [0, 1]$; following a naive approach, one of the elements of the initial partition would be $e \times e = \{(x_1, x_2) \ : \ 0 \leq x_1 \leq 1 \ , \ 0 \leq x_2 \leq 1\}$, related to the location of 2 facilities on e (i.e., $m(e) = 2$). However, $e \times e$ leads to symmetric solutions (for instance, $(0.2, 0.7)$ and $(0.7, 0.2)$ are the same solution since both components belong to the same edge e), making the branch-and-bound algorithm inefficient. In order to avoid this inefficient approach, the set $e \times e$ is replaced by $\{(x_1, x_2) \ : \ 0 \leq x_1 \leq x_2 \leq 1\}$ in the partition.

The subdivision of each element of the partition is done by splitting each (sub)edge by its midpoint, obtaining two new smaller segments for each (sub)edge, namely lower and upper segments. Then, the new elements of the partition are built by replacing each (sub)edge s_i by either its lower or upper segment, s_i^L, s_i^U respectively. In the case of (sub)edges with multiplicity greater than 1, the above-described method is used to avoid symmetric sets. For instance, Figure 1 depicts the subdivision process, for $p = 2$, of the element $\underline{s} = (s_1)$, with $m(s_1) = 2$, identified with the blue colored area of the big square. Then, the subdivision of $\underline{s}$ leads to three new elements, identified with the blue colored area of the small squares.

In order to illustrate the construction of the initial partition and the subdivision process, a simple example is given next.

Example 3.1 *Let us consider the network in Figure 2, where all the edges have length 1, and $p = 2$. The initial partition is obtained by considering all pairs of edges (with repeats) and applying the procedure that has just been described to avoid symmetry when the multiplicity of an edge is greater than one. The six elements of such partition are shown in the following table, where $e_1 = (1, 2)$, $e_2 = (1, 3)$, and $e_3 = (2, 3)$.*

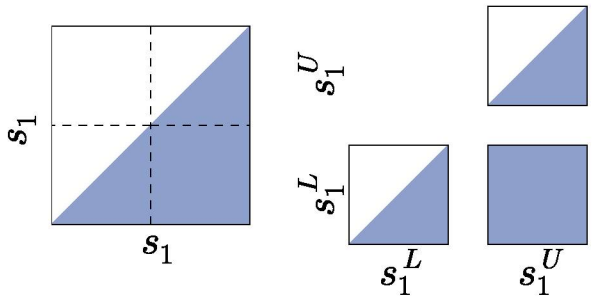

Figure 1: Subdivision process of $\underline{s} = (s_1)$ with $m(s_1) = 2$.

Element	*Multiplicities*	*Description of the element*
$\underline{s}_1 = (e_1)$	$m(e_1) = 2$	$\{(x_1, x_2) : x_1, x_2 \in e_1, 0 \leq x_1 \leq x_2 \leq 1\}$
$\underline{s}_2 = (e_1, e_2)$	$m(e_1) = 1, m(e_2) = 1$	$e_1 \times e_2$
$\underline{s}_3 = (e_1, e_3)$	$m(e_1) = 1, m(e_3) = 1$	$e_1 \times e_3$
$\underline{s}_4 = (e_2)$	$m(e_2) = 2$	$\{(x_1, x_2) : x_1, x_2 \in e_2, 0 \leq x_1 \leq x_2 \leq 1\}$
$\underline{s}_5 = (e_2, e_3)$	$m(e_2) = 1, m(e_3) = 1$	$e_2 \times e_3$
$\underline{s}_6 = (e_3)$	$m(e_3) = 2$	$\{(x_1, x_2) : x_1, x_2 \in e_3, 0 \leq x_1 \leq x_2 \leq 1\}$

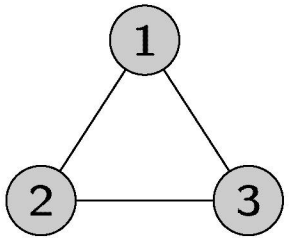

Figure 2: Example network to illustrate the total enumeration algorithm

During the branch-and-bound procedure, the subdivision of each element of the partition with multiplicity equals 1 can be carried out in a straightforward manner. For instance, the subdivision of $\underline{s}_2$ *leads to four new elements in the partition, replacing* $\underline{s}_2$*:*

Element	*Multiplicities*	*Description of the element*
$\underline{s}_{2,1} = (e_{1,1}, e_{2,1})$	$m(e_{1,1}) = 1, m(e_{2,1}) = 1$	$e_{1,1} \times e_{2,1}$
$\underline{s}_{2,2} = (e_{1,1}, e_{2,2})$	$m(e_{1,1}) = 1, m(e_{2,2}) = 1$	$e_{1,1} \times e_{2,2}$
$\underline{s}_{2,3} = (e_{1,2}, e_{2,1})$	$m(e_{1,2}) = 1, m(e_{2,1}) = 1$	$e_{1,2} \times e_{2,1}$
$\underline{s}_{2,4} = (e_{1,2}, e_{2,2})$	$m(e_{1,2}) = 1, m(e_{2,2}) = 1$	$e_{1,2} \times e_{2,2}$

where

$$e_{1,1} = \{x \in e_1 : 0 \leq x \leq \tfrac{1}{2}\} \qquad e_{1,2} = \{x \in e_1 : \tfrac{1}{2} \leq x \leq 1\}$$
$$e_{2,1} = \{x \in e_2 : 0 \leq x \leq \tfrac{1}{2}\} \qquad e_{2,2} = \{x \in e_2 : \tfrac{1}{2} \leq x \leq 1\}$$

When any component of the element to be subdivided has a multiplicity greater than 1, the split must be done following the above-described procedure to avoid symmetric sets. For instance, the subdivision of $\underline{s}_1$ *yields (see Figure 1):*

Element	*Multiplicities*	*Description of the element*
$\underline{s}_{1,1} = (e_{1,1})$	$m(e_{1,1}) = 2$	$\{(x_1, x_2) : x_1, x_2 \in e_{1,1}, 0 \leq x_1 \leq x_2 \leq \frac{1}{2}\}$
$\underline{s}_{1,2} = (e_{1,2}, e_{1,1})$	$m(e_{1,2}) = 1, m(e_{1,1}) = 1$	$e_{1,2} \times e_{1,1}$
$\underline{s}_{1,3} = (e_{1,2})$	$m(e_{1,2}) = 2$	$\{(x_1, x_2) : x_1, x_2 \in e_{1,2}, \frac{1}{2} \leq x_1 \leq x_2 \leq 1\}$

3.2. Superset

A more sophisticated data structure for location problems on networks has been proposed in Blanquero et al. (2015), exploiting together the structure of the combinatorial and continuous part of a covering problem on networks. In order to avoid the enumeration of every possible combination of p edges, Blanquero et al. (2015) propose to construct clusters of (sub)edges, called hereafter *edgesets*, and define a subproblem of (5) over a collection of edgesets called a *superset*.

To be precise, an edgeset is a finite collection of (sub)edges of E; a superset S is any uple of the form $(E_1, p_1; \ldots; E_k, p_k)$, where $E_1, \ldots, E_k$ are disjoint edgesets, p_j are strictly positive integer numbers with

$$\sum_{j=1}^{k} p_j = p,$$

indicating, for each $j = 1, \ldots, k$, that exactly p_j facilities are to be located within the (sub)edges in E_j.

For this data structure, the subproblem to be solved at this stage on superset S has the form

$$\max_{(x_1,\ldots,x_p)\in S} F(x_1, \ldots, x_p)$$

with F defined as in (6), and $(x_1, \ldots, x_p) \in S$ understood as $x_1, \ldots, x_{p_1} \in E_1;\ x_{p_1+1}, \ldots, x_{p_1+p_2} \in E_2;\ \ldots;\ x_{p-p_k+1}, \ldots, x_p \in E_k$.

Supersets will be identified with nodes in the branch-and-bound tree. The root node of the branch-and-bound tree is the original superset $S_0 = (E, p)$. E is first subdivided into a given partition $E^{(1)}, \ldots, E^{(p)}$ of E: we add to the branch-and-bound exploration tree the supersets of the form $(E_1, p_1; \ldots; E_k, p_k)$, where $\{E_1, \ldots, E_k\} \subset \{E^{(1)}, \ldots, E^{(p)}\}$ and $p_1 + \ldots + p_k = p$. Each initial superset can be seen as the result of choosing p elements, not necessarily distinct, from the set $\{E^{(1)}, \ldots, E^{(p)}\}$, and taking p_j as the number of times that E_j appears in the sample; the number of possible elections of this kind corresponds to the concept of *combinations with repetitions* in Combinatorial Theory, and is given by the combinatorial number $\binom{2p-1}{p}$.

First, we need to define how the edges of the network conforming S_0, are split into the partition of p edgesets $E^{(1)}, \ldots, E^{(p)}$. In the first step, E is divided into 2 edgesets by a distance criterion, namely the diameter of the edgeset, defined as the maximum of the minimal distance between each pair of nodes in $V(E)$, the set of nodes that define the edges of E. Then, the nodes giving the diameter are selected as centers of the two new (sub)edgesets. Each edge of the edgeset is assigned to the closest (sub)edgeset, where the distance from an edge to an (sub)edgeset is measured as the distance from the edge to the (sub)edgeset center. In case of tie, the edge is assigned to the (sub)edgeset with the smallest cardinality, or randomly if a new tie arises then. This process is repeated until obtaining p edgesets, selecting the

largest edgeset to be subdivided at each step, where the size of the edgeset is understood as the sum of the (sub)edge lengths; if two edgesets yield the maximum diameter, the edgeset with the biggest cardinality is chosen, whereas a random election is carried out in case of a new tie. An algorithmic description of the whole initialization process is provided in Algorithm 1.

The subdivision of a superset $S = (E_1, p_1; \ldots; E_k, p_k)$ during the branch-and-bound is done by partitioning the largest edgeset E_i. If E_i contains only one (sub)edge, the subdivision is done by bisecting the (sub)edge at its midpoint, otherwise E_i is partitioned to edgesets E_{i_1}, E_{i_2} by its diameter as done in the initial subdivision of S_0. The p_i facilities at E_i must be shared out between E_{i_1} and E_{i_2}, and all the possible combinations in this sharing out of p_i must be considered in order to continue having a partition of the solutions space; therefore, if (n_{i_1}, n_{i_2}) represents the number of facilities assigned to E_{i_1} and E_{i_2}, respectively, with $n_{i_1} + n_{i_2} = p_i$, the subdivision process must take into account the following facilities assignments: $(0, p_i), (1, p_i - 1), (2, p_i - 2), \ldots, (p_i - 2, 2), (p_i - 1, 1), (p_i, 0)$, provided that these pairs make sense (i.e., all their elements are nonnegative). Thus, the following supersets substitute S:

$$S_j = (E_1, p_1; \ldots; E_{i-1}, p_{i-1}; E_{i_j}, p_i; E_{i+1}, p_{i+1}; \ldots; E_k, p_k), j = 1, 2$$

and additionally if $1 < p_i$, for $j = 1, \ldots, p_i - 1$

$$S_{2+j} = (E_1, p_1; \ldots; E_{i-1}, p_{i-1}; E_{i_1}, j; E_{i_2}, p_i - j; E_{i+1}, p_{i+1}; \ldots; E_k, p_k).$$

This means that, in each step, $p_i + 1$ new supersets are created.

Example 3.2 *Let us consider the network $N = (V, E)$ depicted in Figure 3, with all lengths equal to 1, and suppose that $p = 3$ facilities are going to be located. The initial superset is $S_0 = (E, 3)$ and we begin by dividing E into a partition $E^{(1)}, E^{(2)}, E^{(3)}$ as is described next.*

The diameter of E is given by the distance between the nodes 1 and 7, which are considered as centers of two new edgesets. Assigning each edge to its closest center, the new edgesets turn out to be $E^{(1)} = \{(1, 2), (2, 3), (2, 4)\}$ and $E^{(2)} = \{(3, 5), (3, 6), (4, 6), (5, 7), (6, 7)\}$. The sizes of $E^{(1)}$ and $E^{(2)}$ are 3 and 5, respectively, and therefore, $E^{(2)}$ is divided into two new edgesets. The centers $E^{(2)}$ are the nodes 4 and 5, since they provide the diameter of such edgeset, and the assignment of each edge to its closest center yields the follow-

Algorithm 1: Building the initial branch-and-bound exploration tree

▶ `Initialize the list of edgesets`
$L \leftarrow E$
repeat
  • `Find the largest edgeset` E_w^* `in` E
  foreach $E_w \in L$ **do**
    $s(E_w) \leftarrow \sum_{e \in E_w} l_e$
    $E_w^* \leftarrow \text{argmax}_{E_w \in L}\, s(E_w)$
  end
  • `Compute the vertices defining the diameter of` E_w^*
  $(c_1^*, c_2^*) \leftarrow \text{argmax}_{a_i, a_j \in V(E_w^*)}\, d(a_i, a_j)$
  $E_{w,1}^* \leftarrow \emptyset$
  $E_{w,2}^* \leftarrow \emptyset$
  foreach $e \in E_w^*$ **do**
    if $d(e, c_1^*) \leq d(e, c_2^*)$ **then**
      $E_{w,1}^* \leftarrow E_{w,1}^* \cup e$
    else
      $E_{w,2}^* \leftarrow E_{w,2}^* \cup e$
    end
  end
  • `Update` L
  $L \leftarrow L \backslash E_w^* \cup E_{w,1}^* \cup E_{w,2}^*$
until $|L| = p$;

▶ `Generate combinations with repetitions of` $\{1, 2, \ldots, p\}$
$CR \leftarrow \{(a_1, a_2, \ldots, a_p) : a_1 \leq a_2 \leq \ldots \leq a_p, a_i = 1, \ldots, p\}$

▶ `Build supersets and add them to the B&B tree`
$T \leftarrow \emptyset$
foreach $c \in CR$ **do**
  Let $\{u_1, \ldots, u_k\}$ be the set of unique numbers in c.
  Let n_i be the absolute frequency of u_i in c, $i = 1, \ldots, k$.
  $T \leftarrow T \cup (E_{u_1}, n_1; E_{u_2}, n_2; \ldots; E_{u_k}, n_k)$
end

ing subdivision of $E^{(2)}$*:* $E^{(2,1)} = \{(3,6),(4,6)\}$, $E^{(2,2)} = \{(3,5),(5,7),(6,7)\}$*; note that other subdivisions are also possible since the edges* $(3,6)$ *and* $(6,7)$ *are at the same distance from both centers. Renaming the edgesets that have just been worked out, the initial partition of* E *is obtained:*

$$\begin{aligned} E^{(1)} &= \{(1,2),(2,3),(2,4)\} \\ E^{(2)} &= \{(3,6),(4,6)\} \\ E^{(3)} &= \{(3,5),(5,7),(6,7)\} \end{aligned}$$

Once the partition of E *has been computed, we add to the branch-and-bound exploration tree the* $\binom{5}{3} = 10$ *supersets of the form* $(E_1, p_1; \ldots; E_k, p_k)$ *where* $\{E_1, \ldots, E_k\} \subset \{E^{(1)}, E^{(2)}, E^{(3)}\}$ *and* $p_1 + \ldots + p_k = 3$*:*

$$\begin{array}{ll} (E^{(1)}, 3) & (E^{(1)}, 1; E^{(2)}, 2) \\ (E^{(2)}, 3) & (E^{(1)}, 1; E^{(3)}, 2) \\ (E^{(3)}, 3) & (E^{(2)}, 1; E^{(3)}, 2) \\ (E^{(1)}, 2; E^{(2)}, 1) & (E^{(2)}, 2; E^{(3)}, 1) \\ (E^{(1)}, 2; E^{(3)}, 1) & (E^{(1)}, 1; E^{(2)}, 1; E^{(3)}, 1) \end{array}$$

In order to illustrate the subdivision of a superset during the execution of the branch-and-bound algorithm, let us consider the superset $(E^{(1)}, 2; E^{(2)}, 1)$*. The edgeset* $E^{(1)}$ *is choosen for splitting, since it is larger than* $E^{(2)}$*, and divided into the edgesets* $E^{(1,1)} = \{(1,2)\}$ *and* $E^{(1,2)} = \{(2,3),(2,4)\}$ *according to the diameter of* $E^{(1)}$ *(other configurations are also admissible owing to ties). The original superset is then replaced in the branch-and-bound tree by the following supersets:*

$$(E^{(1,1)}, 2; E^{(2)}, 1) \qquad (E^{(1,2)}, 2; E^{(2)}, 1) \qquad (E^{(1,1)}, 1; E^{(1,2)}, 1; E^{(2)}, 1)$$

4. Lower and Upper bounds

A branch-and-bound algorithm requires the calculation of tight upper and lower bounds. In this section we present different bounding approaches for the branch-and-bound used to solve (5). We propose two upper bounds and two lower bounds: IA bound, DC bound as upper bounds, Huff discrete bound (HD) and midpoint bound (MP) as lower bounds. Note that when a superset contains edgesets with $|E_j| = 1 \ \forall j$, it corresponds to a p-uple of (sub)edges. Each p-uple of (sub)edges has its unique superset correspon-

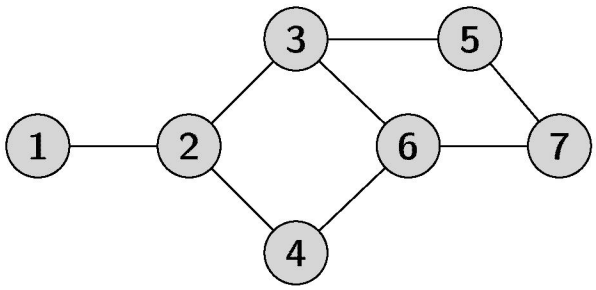

Figure 3: Example network to illustrate the superset approach

dence. Thus, the following bounds are valid for p-uples of edges as well, i.e., for the enumeration approach in Section 3.1.

4.1. Upper bounds

Interval Analysis (IA) and Difference of Convex (DC) bounds have already been proposed for the single-facility case, in which bounds are computed on intervals (subedges). We adapt these bounding strategies to the multifacility case, when bounds of the objective function on supersets are to be computed. The IA bound considers only endpoints of (sub)edges as possible location of facilities. For a superset $S = (E_1, p_1; \ldots; E_k, p_k)$ we obtain the IA bound by replacing in (6) $d(a, x_j)$ by the distance from a to the closest vertex of the edges that belong to E_j, i.e., by

$$d(a, E_j) = \min_{e=[v_1,v_2], e \in E_j} \{d(a, v_1), d(a, v_2)\}.$$

For any $x \in E_j$ it holds that $d(a, E_j) \leq d(a, x)\ \forall j = 1, \ldots, p$. Hence, the following is a valid upper bound for (6):

$$UB^{IA}(S) := \sum_{a \in V} \omega_a \frac{1}{1 + \frac{\beta_a}{\sum_{j=1}^{k} \frac{p_j}{(d(a,E_j))^2}}}.$$

The second upper bound approach is based on a DC bound of the single facility Huff location problem on networks,

$$\max_{x \in [0, l_e],\, e \in E} F_{single}(x)$$

with $F_{single}(x) = \sum_{a \in V} \omega_a \frac{1}{1+\beta_a (d(a,x))^2}$, a particular case of (6) for $p = 1$. This

problem has been already studied in Blanquero et al. (2014). First, for a given edge e of the network, $F_{single}(x)$ is expressed as a difference of convex functions, $F_{single}(x) = \sum_{a \in V}(F_a^+(x) - F_a^-(x))$, namely, its DC decomposition. Then, an upper bound $UB^{DC}_{single}(s)$ for any segment $s \in e,\ e \in E$ with $v_1,\ v_2$ being endpoints of s is defined as

$$UB^{DC}_{single}(s) = \max\{U(v_1), U(v_2)\}$$

with

$$U(x) = \sum_{a \in V}(F_a^+(x) - F_a^-(x_0) - \xi_a(x - x_0))$$

for $\xi_a \in \partial F_a^-(x_0)$ where $\partial F_a^-(x_0)$ denotes the set of subgradients of F_a^- at x_0 (Horst and Pardalos, 1995).
Therefore, it holds that

$$UB^{DC}_{single}(e) \geq F_{single}(x), \forall x \in [0, l_e],\ e \in E. \tag{7}$$

A DC bound over an edgeset E_j is defined as the maximum DC bound of the edges from E_j, i.e.,

$$UB^{DC}(E_j) := \max_{e \in E_j} UB^{DC}_{single}(e) \geq F_{single}(x), \forall x \in [0, l_e],\ e \in E.$$

Given a superset $S = (E_1, p_1; \ldots; E_k, p_k)$, a DC bound of (6) is calculated as

$$UB^{DC}(S) := \sum_{j=1}^{p} p_j \cdot UB^{DC}(E_j) \tag{8}$$

This DC bound is a valid upper bound since it holds that

$$\sum_{j=1}^{p} F_{single}(x_j) = \sum_{j=1}^{p} \sum_{a \in V} \omega_a \frac{1}{1 + \beta_a (d(a, x_j))^2} \tag{9}$$

Since $\frac{1}{(d(a,x_j))^2} \leq \sum_{j=1}^{p} \frac{1}{(d(a,x_j))^2}$, we have:

$$(9) = \sum_{j=1}^{p} \sum_{a \in V} \omega_a \frac{1/(d(a, x_j))^2}{1/(d(a, x_j))^2 + \beta_a} \geq \sum_{j=1}^{p} \sum_{a \in V} \omega_a \frac{1/(d(a, x_j))^2}{\sum_{i=1}^{p} 1/(d(a, x_i))^2 + \beta_a} =$$

$$= \sum_{a \in V} \omega_a \frac{\sum_{j=1}^{p} 1/(d(a,x_j))^2}{\sum_{i=1}^{p} 1/(d(a,x_i))^2 + \beta_a} = \sum_{a \in V} \omega_a \frac{1}{1 + \frac{\beta_a}{\sum_{j=1}^{p} \frac{1}{(d(a,x_j))^2}}} = F(x_1, \ldots, x_p).$$

4.2. Lower bounds

Both of our lower bounding approaches are based on the calculation of the objective function at a feasible solution $(\tilde{x}_1, \ldots, \tilde{x}_p) \in S$. Then, a valid lower bound is given by

$$LB(S) := F(\tilde{x}_1, \ldots, \tilde{x}_p) \leq \max_{(x_1,\ldots,x_p) \in S} F(x_1, \ldots, x_p).$$

Let us now focus on possible feasible solutions. The first lower bound, namely Huff discrete bound (LB^{HD}), is a greedy procedure based on solving iteratively p times the single facility Huff location problem at the vertices of the edges of the superset. For a given superset $S = (E_1, p_1; \ldots; E_k, p_k)$, let $\tilde{x}_1$ be the optimal solution of a single facility Huff location problem on the vertices of the (sub)edges of E_1. In the next step, we will consider that a facility is already located at $\tilde{x}_1$, and will locate $\tilde{x}_2$ solving the single facility Huff location problem at the vertices of the edges of the corresponding edgeset. In the last step, we will choose location $\tilde{x}_p$ as the optimal solution of the single facility Huff location problem on the vertices of the edges of E_k, considering that $p-1$ facilities are already located at $\tilde{x}_1, \ldots, \tilde{x}_{p-1}$. Since only vertices are considered as candidates, each step of the greedy procedure is executed by complete enumeration of the candidate points.

The second lower bound, namely the midpoint bound LB^{MP}, is calculated by randomly choosing an edge from an edgeset E_j, and locating p_j facilities at its midpoint $\forall j \leq k$.

5. Computational results

The approaches described in Sections 3 and 4 were implemented in Fortran and executed on an Intel Core i7 computer with 16.00 Gb of RAM memory. The solutions were found within an accuracy of 10^{-3}.

We tested the two approaches on a battery of 21 networks, whose characteristics are shown in Table 1. The first 9 networks, referred to as *group NT1* in the sequel, are artificial networks generated as described in Averbakh et al.

(2014). The following 5 networks (*group NT2*) are proposed for p-median problems in Beasley (1990) and also used in Berman et al. (2011). Finally, the last 7 networks (*group NT3*) are taken from (Corberán and Sanchis, 2007; Reinelt, 1991).

The number r of existing facilities is set as $r = 10\%$ of the number of edges of the network, $|E|$. Each instance is obtained by randomly and independently generating the demands (each vertex of the network is assumed to have a demand uniformly distributed in the interval $(0, 1)$ and in $(0, 20)$ for the artificial networks from Averbakh et al. (2014)) and the location of the existing facilities. To generate the latter, r edges are randomly chosen with replacement; on each selected edge, the location of the facility is generated following a uniform distribution.

For each instance, the computer program written to solve the problem here considered runs until one of the following conditions is fulfilled:

- An optimal solution is found.
- The size limit (10^8) of the branch-and-bound tree is exceeded.
- The cpu time exceeds six hours (21600 seconds).

Tables 2 and 3 show a comparison between the two branching rules: total enumeration (Section 3.1) and superset (Section 3.2), and the different bounding approaches: IA bound, IA bound with DC bound (IA+DC), midpoint evaluation bound (MP), and the combination of Huff discrete bound with midpoint evaluation (HD+MP). Results for DC bound as the only upper bound are not reported because they were systematically outperformed by the other upper bounds; due to this reason, the outcomes of HD bound are also omitted. The results for the combinations of lower and upper bounds are shown in four blocks of columns. The first column of each block shows the average maximum size of the branch-and-bound tree (MaxList) in each group of networks during the execution of the algorithm, whereas the second column reports the average CPU time in seconds. In what follows, the pair (LB,UB) will denote the combination of given lower and upper bounds used in the algorithm.

We start with the analysis of $p = 2$. All strategies are able to solve the problem on the 21 networks in less than an average time of 2 seconds in all the groups. For the enumerative approach, the best results in terms of cpu time are obtained when (MP,IA) are used. If we focus on MaxList,

(HD+MP,IA+DC) provides the best election. A good balance between memory requirement and cpu time is obtained when one considers the combination (MP,IA+DC). Regarding the superset approach, the best results are provided by (HD+MP,IA+DC), although these results are only slightly better that those obtained when the combination (MP,IA+DC) is used. The comparison of both methodologies shows that the superset approach is the fastest one and enumeration achieves the best MasList size as a rule.

For $p = 3$, using supersets one achieves the best computing time, whilst enumeration provides the best Maxlist result. For the superset approach, the best combination turns out to be (HD+MP,IA+DC) even though similar results are provided by (MP,IA+DC) and, to a lesser extent, by (HD+MP,IA) and (MP,IA) (the use of the DC upper bound seems to lead to better results). In the case of the enumeration approach, the choice of the upper bound affects both the MaxList size and the cpu time. Using IA+DC bound causes an important reduction of the MaxList size compared to using only IA bound, especially in the group NT3, and a moderate rise of cpu time. On the other hand, the use of HD+MP as lower bound reduces on average the MaxList size by half and increases threefold the cpu time. In terms of memory requirement, the best bounds choice is (HD+MP,IA+DC), and when cpu time is the relevant measurement, the best choice is (MP,IA), which is the least efficient combination in terms of memory use.

The case $p = 4$ is considered now. The enumeration approach solves 19 networks when IA+DC is used as lower bound and 20 networks with IA bound. Supersets solves 20 networks regardless of the bounds used. The $RAT195G$ network is not solved in any case and its gap ranges from 9.93% when IA+DC is used as upper bound, to 15.94% with IA bound. In terms of time, the best results are provided by the superset approach, with HD+MP the best choice as lower bound, while the choice of upper bound does not make big difference; as in the case $p = 3$, the best cpu times are given by the combination (HD+MP,IA+DC), which has a similar performance to (MP,IA+DC) and (HD+MP,IA). On the other hand, the enumerative approach is the most efficient when MaxList size is the target measurement, being (HD,IA) and (HD+MP,IA) the best combinations in that case. For the superset approach, the use of IA as upper bound minimizes MaxList size.

Finally, we analyze results for $p = 5$. Using enumeration we are able to solve the problem on 8 networks in NT1 using (HD+MP,IA+DC), (MP,IA+DC) and (MP,IA). Only when the combination (HD+MP,IA) was used, the solu-

tion of all the problems in NT1 could be obtained; none of the problems in NT2 and NT3 were solved and the gap ranged from 2.37% to 59.72%. The number of problems solved with the superset approach is seven in all cases, with a gap ranging from 4.13% to 48.90%. In terms of computing time, the combination (MP,IA) gives the best computational times in the enumerative approach, although these ones are clearly outperformed by the superset approach, where the best results are obtained when the HD+MP lower bound is considered, regardless of the upper bound used. As for the MaxList size, both approaches provide similar results, slightly better when supersets are used. If one seeks a proper balance between memory requirements and cpu time, the best choice for the superset approach is (HD+MP,IA) or (HD+MP,IA+DC), and (MP,IA) for the enumerative approach.

In summary, it can be said that, to a certain extent, both approaches are comparable for $p = 2, 3$ while for $p = 4, 5$ the superset approach outperforms the enumeration approach. When faced with the choice of the best upper bound, using IA+DC bound as upper bound is the best choice for both approaches and all values of p; if we focus on cpu time exclusively, IA is the best upper bound for the enumerative approach. In terms of lower bound, we found that using MP is, in general, the best choice, even though HD+MP provides slightly better results than MP for the superset approach.

For the sake of completeness, the influence of the number of existing facilities has been studied using the superset approach with IA+DC as upper bound and MP as lower bound. As in the previous computational experiments, the number r of existing facilities was set to a certain percentage (5, 10, 20, 40, 60, 80, and 90) of the number of edges of the network in each run of the algorithm; the demands and the location of the existing facilities were chosen as was described above for the general experimentation. For each group of networks and each number p of new facilities, the averages of the two indicators previously considered (maximum size of the branch-and-bound tree and CPU time) were calculated; the results are graphically depicted in Figures 4 and 5. For $p = 4$ and $r = 80\%$ the results are omitted because only one problem could be solved in that case. For $p = 5$, only results for smallest networks (group NT1) are shown in Figures 4 and 5 since the size limit of the branch-and-bound tree was exceeded in the remaining problems; due to the same reason, the result corresponding to $r = 90\%$ is not displayed for the same value of p. The number of existing facilities r has influence over both the Max List size and the cpu time, especially when the size of the networks

Table 1: Properties of the networks taken from Averbakh et al. (2014); Beasley (1990); Corberán and Sanchis (2007); Reinelt (1991)

Group	Network	nodes	edges
NT1	art1	20	38
	art2	20	43
	art3	20	51
	art4	30	56
	art4	30	71
	art5	30	84
	art7	40	74
	art8	40	95
	art9	40	115
NT2	pmed1	100	196
	pmed2	100	191
	pmed3	100	196
	pmed4	100	194
	pmed5	100	194
NT3	KROB150G	150	296
	KROA150G	150	297
	PR152G	152	296
	RAT195G	195	336
	KROB200G	200	386
	KROA200G	200	392
	TS225G	225	306

grows; the larger r is, the larger the MaxList size and the cpu time are. A deeper analysis reveals that the relative increment originated by changes in r is more pronounced in group NT2, followed by group NT3 as a rule. For instance, the MaxList size for $p = 4$, group NT2 and $r = 90\%$ is 7.6 times the corresponding value for $r = 5\%$. Generally speaking, this effect is smoother in terms of cpu time; in the previous example, the cpu time for $r = 90\%$ is 3.3 times the corresponding value for $r = 5\%$.

6. Conclusions

In this paper we have addressed the p-facility Huff location problem on networks. We propose two branch-and-bound based approaches and show results for $p \leq 5$. Computational results show that both division approaches are able to solve problems of rather realistic size up to $p = 4$ facilities while for $p = 5$ only small problems are solved. For small values of p, both approaches are comparable, and when the number of facilities increases, the superset approach outperforms the enumeration approach. We conclude with four promising extensions.

As shown in Section 5, for high values of p and for both approaches, some problems remain unsolved because the MaxList size limit is reached. It could be interesting to design a heuristic approach able to reduce the number of elements of the partition, and exploit the benefits of the branch-and-bound tree evolution.

As second extension, we have considered throughout the paper that the attractiveness of the new facilities is fixed in advance. Considering it as another decision variable, as, among others, in Fernández et al. (2007); Plastria and Carrizosa (2004); Tóth et al. (2009) calls for designing new upper bounds. While both IA and DC bounds can be adapted to this more general context, the empirical performance of such new bounds is unknown.

Third, the branch and bound strategies used in this paper can also be applied to different p-facility location problems on networks, such as the p-median problem with continuous demand on a network (Blanquero and Carrizosa, 2013).

Finally, parallelization techniques deserve further study. Parallelizing the approach can solve the problem of reaching the MaxList size limit and may reduce the computational cost linearly, which will definitely lead to solving the problem for higher values of p.

References

Averbakh, I., Berman, O., Krass, D., Kalcsics, J., Nickel, S., 2014. Cooperative covering problems on networks. Networks 63 (4), 334–349.

Beasley, J., 1990. OR-library: Distributing test problems by electronic mail. Journal of the Operational Research Society 41, 1069–1072.

Table 2: Average maximum branch-and-bound tree size and running times for the enumeration approach

Enumeration								
	Upper bound	IA				IA+DC			
	Lower bound	MP		HD+MP		MP		HD+MP	
p	Networks group	MaxList	Time	MaxList	Time	MaxList	Time	MaxList	Time
2	NT1	1614	0.02	1114	0.02	1137	0.02	780	0.03
	NT2	2752	0.15	1673	0.30	961	0.19	582	0.36
	NT3	5555	0.68	3568	1.47	1048	0.93	499	1.80
3	NT1	62.082	1.17	32828	1.61	64165	1.28	43584	2.10
	NT2	213728	18.05	123503	42.56	149756	21.23	96958	52.34
	NT3	497173	98.99	262876	352.65	128531	151.25	56581	445.11
4	NT1	1918016	47.20	829900	89.65	2308857	83.00	1563900	140.88
	NT2	11948825	1244.77	6200215	4609.43	15424382	2282.46	10722153	5638.55
	NT3	26613106	8831.86	14621750	10789.24	19904191	17733.14	16746091	17872.20
5	NT1	27146230	1319.57	18025866	3146.07	37477920	2871.15	26435772	4084.51
	NT2	-	-	-	-	-	-	-	-
	NT3	-	-	-	-	-	-	-	-

Table 3: Average maximum branch-and-bound tree size and running times for the superset approach

Supersets									
	Upper bound	IA				IA+DC			
	Lower bound	MP		HD+MP		MP		HD+MP	
p	Networks group	MaxList	Time	MaxList	Time	MaxList	Time	MaxList	Time
2	NT1	1883	0.03	1703	0.02	1771	0.03	1590	0.03
	NT2	4407	0.18	3990	0.17	4130	0.18	3635	0.17
	NT3	7819	0.55	7680	0.50	5203	0.47	5214	0.46
3	NT1	74025	1.12	65659	1.10	73998	1.10	65647	1.09
	NT2	349935	16.01	311701	15.65	347219	16.06	308682	15.56
	NT3	757185	57.81	744442	55.73	564898	50.72	555053	50.50
4	NT1	2314628	41.00	1946402	39.54	2314629	41.41	1946402	41.52
	NT2	15474835	759.09	13194372	741.53	15470797	751.61	13164446	740.38
	NT3	29354658	3893.07	29292169	2751.57	28697363	2706.62	28634874	2659.87
5	NT1	29737308	581.31	23080325	535.79	29737310	563.21	23080326	538.78
	NT2	-	-	-	-	-	-	-	-
	NT3	-	-	-	-	-	-	-	-

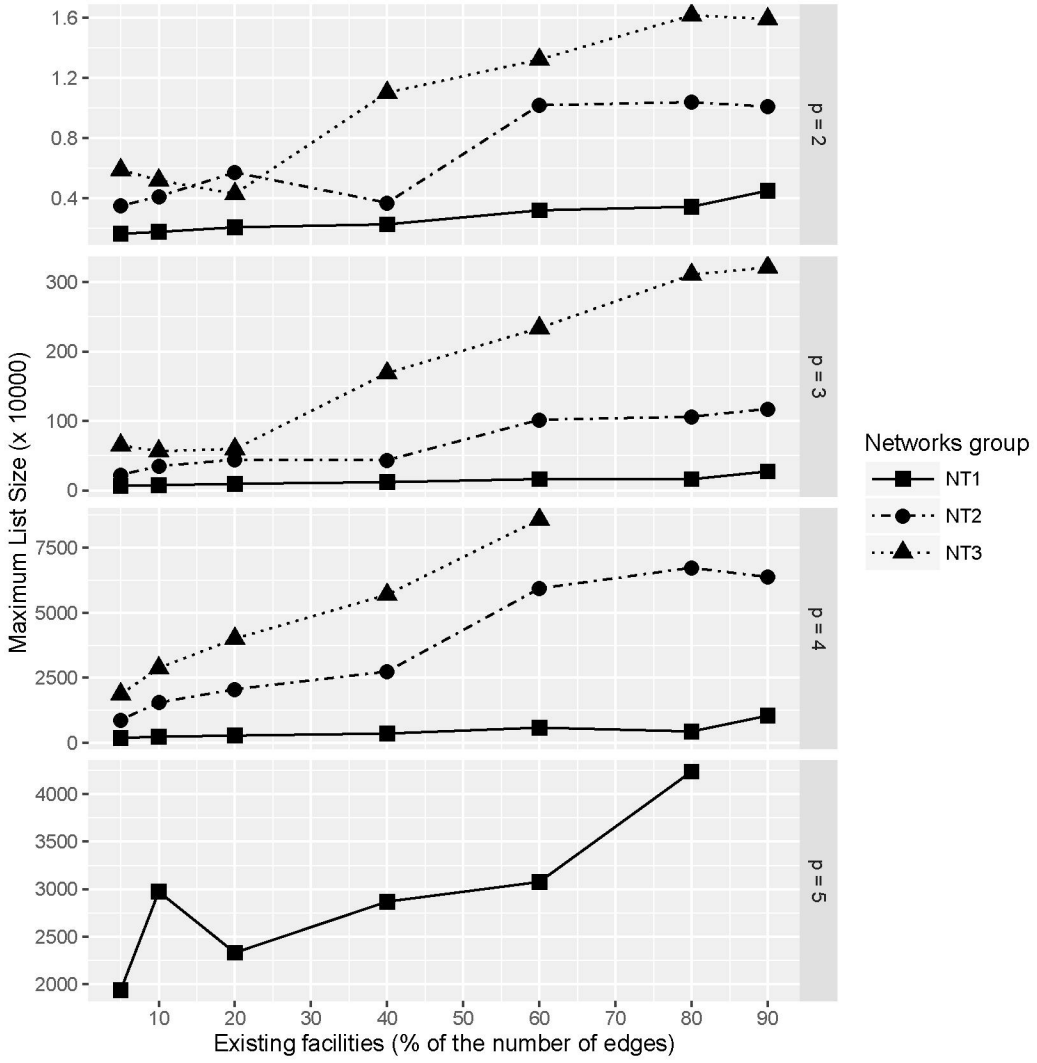

Figure 4: Maximum size of the B&B tree size when varying the number of existing facilities

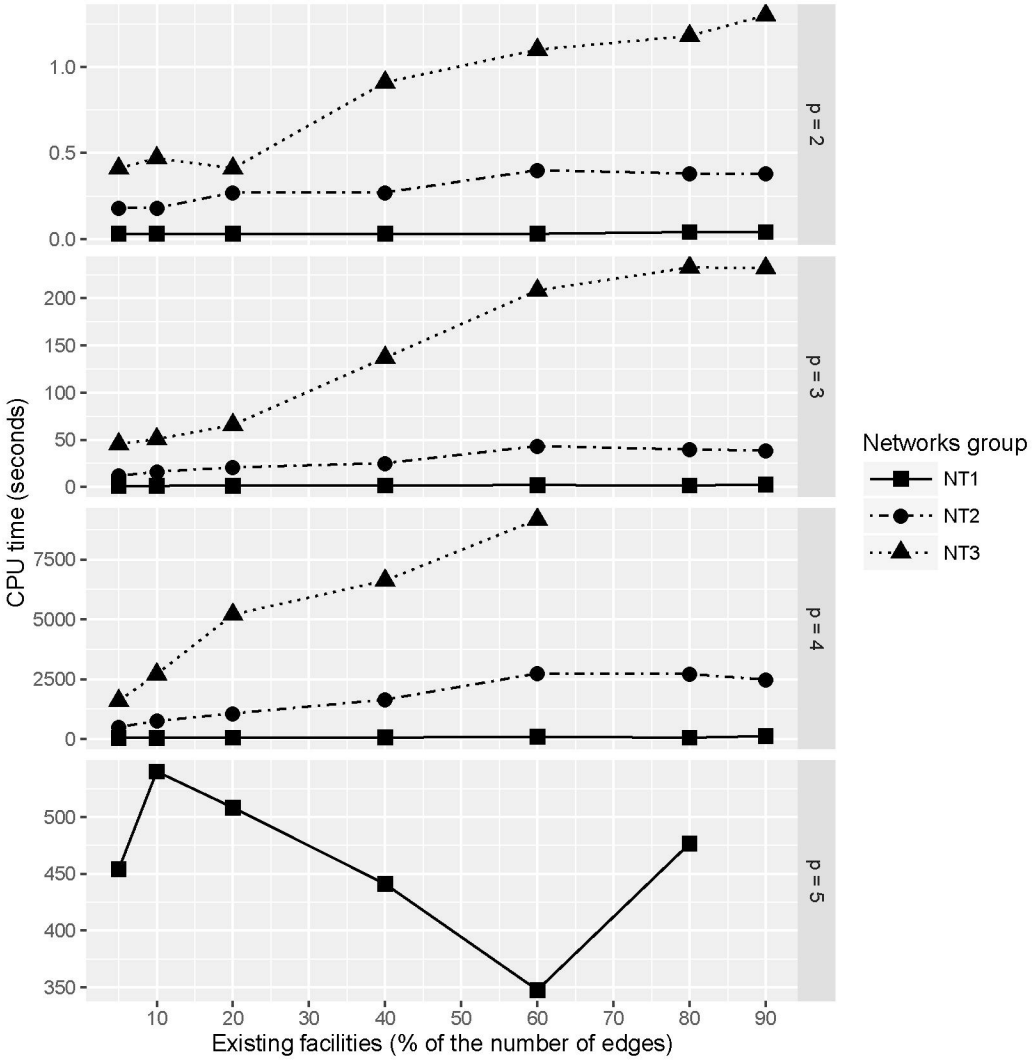

Figure 5: CPU time when varying the number of existing facilities

Berman, O., Drezner, Z., Krass, D., 2011. Big segment small segment global optimization algorithm on networks. Networks 58 (1), 1–11.

Bertsekas, D., 1998. Network Optimization: Continuous and Discrete Models. Athenas Scientific, Belmont, Mass.

Blanquero, R., Carrizosa, E., 2009. Continuous location problems and big triangle small triangle: Constructing better bounds. Journal of Global Optimization 45 (3), 389–402.

Blanquero, R., Carrizosa, E., 2013. Solving the median problem with continuous demand on a network. Computational Optimization and Applications 56 (3), 723–734.

Blanquero, R., Carrizosa, E., Nogales-Gómez, A., Plastria, F., 2014. Single-facility Huff location problems on networks. Annals of Operations Research 222 (1), 175–195.

Blanquero, R., Carrizosa, E., Tóth, B., 2015. Maximal covering location problems on networks with regional demand. Omega.
URL http://dx.doi.org/10.1016/j.omega.2015.11.004

Corberán, A., Sanchis, J., 2007. A branch & cut algorithm for the windy general routing problem and special cases. Networks 49 (4), 245–257.

Drezner, T., 1998. Location of multiple retail facilities with limited budget constraints in continuous space. Journal of Retailing and Consumer Services 5 (3), 173–184.

Drezner, T., Drezner, Z., 2004. Finding the optimal solution to the Huff competitive location model. Computational Management Science 1 (2), 193–208.

Drezner, T., Drezner, Z., Salhi, S., 2002. Solving the multiple competitive facilities location problem. European Journal of Operational Research 142 (1), 138–151.

Eiselt, H., Laporte, G., Thisse, J.-F., 1993. Competitive location models: A framework and bibliography. Transportation Science 27 (1), 44–54.

Fernández, J., Pelegrín, B., Plastria, F., Tóth, B., 2007. Solving a Huff-like competitive location and design model for profit maximization in the plane. European Journal of Operational Research 179 (3), 1274–87.

Ghosh, A., McLafferty, S., Craig, C., 1995. Multifacility retail networks. In: Drezner, Z. (Ed.), Facility Location. A Survey of Applications and Methods. Springer, New York, pp. 301–330.

Horst, R., Pardalos, P. (Eds.), 1995. DC Optimization: Theory, Methods and Algorithms, Handbook of Global Optimization. Kluwer Academic Publishers, Dordrecht, Holland.

Hotelling, H., 1929. Stability in competition. Economic Journal 39 (153), 41–57.

Huff, D., 1964. Defining and estimating a trading area. Journal of Marketing 28 (3), 34–38.

Huff, D., 1966. A programmed solution for approximating an optimum retail location. Land Economics 42 (3), 293–303.

Labbé, M., Peeters, D., Thisse, J., 1995. Location on Networks. In: et al., M. B. (Ed.), Handbooks in operations research and management science. Vol. 8. Elsevier, Amsterdam, pp. 551–624.

Okabe, A., Kitamura, M., 1997. A computational method for market area analysis on a network. Location Science 5 (3), 198–198.

Okabe, A., Okunuki, K.-I., 2001. A computational method for estimating the demand of retail stores on a street network and its implementation in GIS. Transactions in GIS 5 (3), 209–220.

Okunuki, K.-I., Okabe, A., 2002. Solving the Huff-based competitive location model on a network with link-based demand. Annals of Operations Research 111 (1-4), 239–252.

Plastria, F., 2001. Static competitive facility location: An overview of optimisation approaches. European Journal of Operational Research 129 (3), 461–470.

Plastria, F., Carrizosa, E., 2004. Optimal location and design of a competitive facility. Mathematical programming 100 (2), 247–265.

Redondo, J. L., Fernández, J., García, I., Ortigosa, P. M., 2009a. Parallel algorithms for continuous multifacility competitive location problems. Journal of Global Optimization 50 (4), 557–573.

Redondo, J. L., Fernández, J., García, I., Ortigosa, P. M., 2009b. Solving the multiple competitive facilities location and design problem on the plane. Evolutionary Computation 17 (1), 21–53.

Reinelt, G., 1991. TSPLIB - A traveling salesman problem library. ORSA Journal on Computing 3 (4), 376–384.

Roksandić, S., Carrizosa, E., Urošević, D., Mladenović, N., 2012. Solving multifacility Huff location models on networks using variable neighborhood search and multi-start local search metaheuristics. Electronic Notes in Discrete Mathematics 39 (1), 121 – 128.

Tóth, B., Fernández, J., Pelegrín, B., Plastria, F., 2009. Sequential versus simultaneous approach in the location and design of two new facilities using planar huff-like models. Computers & Operations Research 36 (5), 1393–1405.