Solving a Huff-like Stackelberg problem on networks

Boglárka G.-Tóth* and Kristóf Kovács†

Abstract

This work deals with a Huff-like Stackelberg problem, where the leader facility wants to decide its location so that its profit is maximal after the competitor (the follower) also built its facility. It is assumed that the follower makes a rational decision, maximizing their profit. The inelastic demand is aggregated into the vertices of a graph, and facilities can be located along the edges.

Though the problem is considered very hard computationally we give a Branch and Bound algorithm that can find the solution in reasonable time on medium networks, and thus contributing to the scientific achievement of the field.

1 Introduction

The Huff-like single facility location problem in [1] deals with the location decision of a firm trying to build a new facility maximizing the market share of the new facility using probabilistic customer choice. The problem was given by a network, a graph with the nodes being demand points and facilities positioned along its edges. A branch and bound algorithm was proposed with upper bounds given by interval analysis and DC decomposition.

Sáiz et.al. [6] solved a Huff-like Stackelberg problem on the plane. The attraction and market share is defined in the same way as in [1], but of course with euclidean distance and facilities being anywhere in the convex hull of the demand points. The planar space inherently makes the problem more difficult compared to the same problem on a network, although in this work the zero-sum property of the objectives helped for solving the problem. Other examples of works on Huff-like Stackelberg problems are the heuristic methods of Drezner and Drezner in [2], and a bi-level approach in [4] by Küçükaydin at al.

In this work we aim to solve the Stackelberg problem on networks maximizing the profit both for the leader and for the follower. In order to find the global

*Corresponding author: Boglárka G.-Tóth, Budapest University of Technology and Economics, Hungary. Email: bog@math.bme.hu

†Budapest University of Technology and Economics, Hungary. Email: kkovacs@math.bme.hu

optimum of the leader problem with ϵ accuracy we design a Branch and Bound algorithm. The problem differs from the previously shown Stackelberg problem [6] not only by having the location space be a network, but by the facilities taking operational costs into account. The latter makes the problem significantly harder, since the objective functions do not have the zero-sum property. In the next section we formulate the model of the problem, then in Section 3 and 4 describe the bounds and solution methods for the follower and leader problem respectively. In Section 5 computational results are shown for small and medium sized networks. Finally, in Section 6 we conclude the paper and indicate future lines of research.

2 Model

Let us now introduce formally the problem under consideration. The distance used on the network is the following. Given a network $N = \langle V, E \rangle$, where each $e \in E$ refers to the edge with end points a_i and $a_j \in V$ its length denoted by l_e. Thus we denote any $x \in [0, l_e]$ by the point in the edge e at distance x of a_i and distance $l_e - x$ of a_j.

The demand is concentrated at the vertices of N, where each $a \in V$ has a buying power ω_a associated with it. The function $d_a(x)$ gives the distance between demand point a and facility x. Assuming that x is located on edge e with end points a_i and a_j, it is calculated as follows

$$d_a(x) = \min\{x + d(a_i, a), l_e - x + d(a_j, a)\}$$

where $d(a_i, a)$ is the length of the shortest path from demand point a_i to a on the network.

Let us introduce the following notations.

Indices

i	indices of users (vertices) ($i = 1 \dots n$)
j	indices of already existing facilities ($j = 1, \dots, k$ belongs to the leader, $j = k+1 \dots m$ belongs to the competitors)
l, f	index of the leader, and the follower

Variables

x_l	position of the leader's new facility
x_f	position of the follower's new facility

Data

a	demand point
x_j	position of facility j
q_j	quality of facility j
q_l	quality of the leader's new facility
q_f	quality of the follower's new facility
ω_a	buying power of a

Miscellaneous

$d_a(x_j)$	distance of demand point a and facility j
φ_a	positive non-decreasing function
$q/\varphi_a(x)$	the attraction a feels for facility at x with quality q
ψ	positive non-decreasing function
$M_l(x_l, x_f)$	market share captured by the leader
$M_f(x_l, x_f)$	market share captured by the follower
$G(x, q)$	operational cost of facility with quality q at x

In a competitive environment it is usual to assume that both firms are already present on the market, owning a number of facilities. The leader's existing facilities will be denoted by x_j $j = 1 \dots k$ and to the follower's by x_j $j = k+1 \dots m$, with their fixed qualities q_j for $j = 1, \dots m$. The location of the leader's new facility is denoted by x_l its quality by q_l, similarly we have x_f and q_f for the follower.

The market share captured by the leader (with new facility at x_l) after the follower locates at x_f is

$$M_l(x_l, x_f) = \sum_{a \in V} \omega_a \frac{q_l/\varphi_a(d_a(x_l)) + \sum_{j=1}^{k} q_j/\varphi_a(d_a(x_j))}{q_l/\varphi_a(d_a(x_l)) + q_f/\varphi_a(d_a(x_f)) + \sum_{j=1}^{m} q_j/\varphi_a(d_a(x_j))},$$

while the market share captured by the follower is

$$M_f(x_l, x_f) = \sum_{a \in V} \omega_a \frac{q_f/\varphi_a(d_a(x_f)) + \sum_{j=k+1}^{m} q_j/\varphi_a(d_a(x_j))}{q_l/\varphi_a(d_a(x_l)) + q_f/\varphi_a(d_a(x_f)) + \sum_{j=1}^{m} q_j/\varphi_a(d_a(x_j))}.$$

The function φ is a positive nondecreasing function on non-negative values. The usual choice is $\varphi(t) = t^\lambda$, where $\lambda = 2$ gives the so called gravitational model. Both firms are assumed to have renting and/or operational costs depending on their proximity to the demand points. Locations near highly populated areas are likely to be more expensive. We suppose the operational cost $G(x, q)$ of a facility at x, with quality q is

$$G(x, q) = \sum_{a \in V} \omega_a \frac{q}{\psi(d_a(x))},$$

where ψ is a similar function to φ, though the two should not be the same. Thus the profit of the two firms are

$$F_l(x_l, x_f) = M_l(x_l, x_f) - G(x_l, q_l), \quad F_f(x_l, x_f) = M_f(x_l, x_f) - G(x_l, q_l).$$

Using the previous functions we can formulate the objective of the leader problem as

$$\begin{aligned} \max_{x_l \in E} \quad & F_l(x_l, x_f^*) \\ \text{s.t.} \quad & x_f^* = \operatorname*{argmax}_{x_f \in E} F_f(x_l, x_f) \end{aligned}$$

Naturally the objective of the follower problem for a given x_l is

$$\max_{x_f \in E} F_f(x_l, x_f).$$

In the following sections we will propose an algorithm to solve both the follower and the leader problem.

3 Follower problem

In this section we describe a Branch and Bound algorithm for the follower problem on networks taking operational costs also into account. The largest set we considered for a subproblem is an edge, making the branching rule, and bound calculations easier. Let us start by describing the estimation of the bounds.

3.1 Lower Bound

The lower bound for the objective of the follower and a fixed leader position is obtained by taking the best objective value at a finite set of feasible solutions $\{x_f^1, x_f^2, \dots x_f^r\}$, i.e.

$$z^L = \max\{F_f(x_l^*, x_f^1), F_f(x_l^*, x_f^2), \dots F_f(x_l^*, x_f^r)\}$$

where x_l^* is a given location of the leader.

3.2 Upper bound

The calculation of the upper bound is done in two parts, first we formulate an upper bound for the market share, then in a similar way we obtain a lower bound for the operational cost, the difference of these two bound will be a correct upper bound on the profit. We considered two methods for determining the upper bound of the market share:

1. Using interval arithmetic, underestimating the distance from the demand points by assuming the follower delivers from its entire segment.

2. Taking advantage of the market share being DC and calculating a DC upper bound.

In the first case, by underestimating the follower's distance to every demand point we overestimate its attraction, thus overestimating its market share. This means the follower delivers from its whole segment, so any point in the segment is zero distance away from every other point of the segment. The computation of it is very cheap, for every demand point two calculations have to be made, one on each extreme point of the segment. This is equivalent to the upper bound given by the natural inclusion of M_f.

The following is the calculation of the DC upper bound. Let us use the following notations for the attraction of the existing facilities owned by the leader, the follower and both of them, respectively:

$$\beta_a^l = \sum_{j=1}^{k} q_j/\varphi_a(d_a(x_j)), \quad \beta_a^f = \sum_{j=k+1}^{m} q_j/\varphi_a(d_a(x_j)), \quad \beta_a = \beta_a^l + \beta_a^f$$

Using the above notations, and assuming $\varphi_a(d) = d_a^\lambda$, we can write the market share of the follower for demand point a as follows:

$$\begin{aligned} m_a(x_f) &= \frac{q_f/d_a^\lambda(x_f) + \beta_a^f}{q_f/d_a^\lambda(x_f) + q_l/d_a^\lambda(x_l) + \beta_a} \\ &= \frac{q_f + \beta_a^f d_a^\lambda(x_f)}{q_f + (q_l/d_a^\lambda(x_l) + \beta_a)\, d_a^\lambda(x_f)} \\ &= \frac{1 + \beta_a^f d_a^\lambda(x_f)/q_f}{1 + (q_l/d_a^\lambda(x_l) + \beta_a)\, d_a^\lambda(x_f)/q_f} \end{aligned}$$

Introducing the notation $\gamma_a = \frac{q_l/d_a^\lambda(x_l)+\beta_a}{q_f}$, we can simplify it further to

$$\begin{aligned} &= \frac{1 + \beta_a^f d_a^\lambda(x_f)/q_f}{1 + \gamma_a d_a^\lambda(x_f)} \\ &= \frac{1 + \frac{\beta_a^f}{\gamma_a q_f}\gamma_a d_a^\lambda(x_f)}{1 + \gamma_a d_a^\lambda(x_f)} \end{aligned}$$

Finally denoting by $\alpha_a = \frac{\beta_a^f}{\gamma_a q_f} = \frac{\beta_a^f}{q_l/d_a^\lambda(x_l)+\beta_a} < 1$, it leads to

$$\begin{aligned} &= \frac{1}{1 + \gamma_a d_a^\lambda(x_f)} + \alpha_a \frac{\gamma_a d_a^\lambda(x_f)}{1 + \gamma_a d_a^\lambda(x_f)} \\ &= \frac{1}{1 + \gamma_a d_a^\lambda(x_f)} + \alpha_a \left(1 - \frac{1}{1 + \gamma_a d_a^\lambda(x_f)}\right) \\ &= \frac{1}{1 + \gamma_a d_a^\lambda(x_f)} + \alpha_a - \alpha_a \frac{1}{1 + \gamma_a d_a^\lambda(x_f)} \\ &= \alpha_a + (1 - \alpha_a)\frac{1}{1 + \gamma_a d_a^\lambda(x_f)} \end{aligned}$$

We will show that the market share of the follower $M_f(x_l, x_f)$ is DC by giving the DC decomposition of every $m_a(x_f)$ function. Consider $h_a(t) = \frac{1}{1+\gamma_a t^\lambda}$, the following DC decomposition of h_a is known:

$$c_a = \left(\frac{\lambda - 1}{(1+\lambda)\gamma_a}\right)^{\frac{1}{\lambda}}$$

$$h_a^+(t) = \begin{cases} h_a(c_a) + h_a'(c_a)(t - c_a) & \text{if } t \le c_a \\ h_a(t) & \text{if } t > c_a \end{cases}$$

$$h_a^-(t) = \begin{cases} h_a(c_a) + h_a'(c_a)(t - c_a) - h_a(t) & \text{if } t \le c_a \\ 0 & \text{if } t > c_a \end{cases}$$

Using the following proposition, we can get a DC decomposition for $m_a(x_f)$:

Statement 3.1. (Blanquero et al. [1]) *Let $I \subset \mathbf{R}$ be an interval. Let $d : I \to \mathbf{R}$ be a concave function on I, and let $g : \mathbf{R} \to \mathbf{R}$ be DC, with a DC decomposition given by $g(x) = g^+(x) - g^-(x)$, with both g^+ and g^- non-increasing functions. Then, the function $f : I \to \mathbf{R}$ defined as $g(d(x))$ is DC on I and a DC decomposition is given by $f(x) = f^+(x) - f^-(x)$ where $f^+(x) = g^+(d(x))$ and $f^-(x) = g^-(d(x))$.*

The DC decomposition of $m_a(x_f)$ is

$$m_a(x_f) = \alpha_a + (1 - \alpha_a)h_a(d_a(x_f))$$

$$m_a^+(x_f) = \begin{cases} \alpha_a + (1 - \alpha_a)\left[h_a(c_a) + h_a'(c_a)(d_a(x_f) - c_a)\right] & \text{if } d_a(x_f) \le c_a \\ \alpha_a + (1 - \alpha_a)h_a(d_a(x_f)) & \text{if } d_a(x_f) > c_a \end{cases}$$

$$m_a^-(x_f) = \begin{cases} \alpha_a + (1 - \alpha_a)\left[h_a(c_a) + h_a'(c_a)(d_a(x_f) - c_a) - h_a(d_a(x_f))\right] & \\ & \text{if } d_a(x_f) \le c_a \\ 0 & \text{if } d_a(x_f) > c_a \end{cases}$$

$$m_a(x_f) = m_a^+(x_f) - m_a^-(x_f)$$

Thus we have a DC decomposition for $M_f(x_l, x_f) = M_f^+(x_l, x_f) - M_f^-(x_l, x_f)$, where

$$M_f^+(x_l, x_f) = \sum_{a \in V} \omega_a m_a^+(x_f)$$

$$M_f^-(x_l, x_f) = \sum_{a \in V} \omega_a m_a^-(x_f)$$

To construct an upper bound on M_f we need the upper bound of M_f^+ and the lower bound of M_f^-. The upper bound of M_f^+ is easily obtained, let us assume that the segment on which the bound is to be found has end points y_1, y_2, then the upper bound of M_f^+ is

$$UB(M_f^+) = \max\{M_f^+(y_1), M_f^+(y_2)\}.$$

To estimate the lower bound of M_f^- we need the first derivative of M_f^-

$$(M_f^-(x_f))' = \sum_{a \in V} \omega_a (m_a^-(x_f))'$$

$$(m_a^-(x_f))' = \begin{cases} (1 - \alpha_a)\left[h_a'(c_a)d_a'(x_f) - d_a'(x_f)h_a'(d_a(x_f))\right] & \text{if } d_a(x_f) \le c_a \\ 0 & \text{if } d_a(x_f) > c_a \end{cases}$$

$$h'_a(t) = -\frac{\gamma_a \lambda t^{\lambda-1}}{(1+\gamma_a t^\lambda)^2}$$

Finally d_a is a piecewise linear function with the pieces having a gradient of ± 1. The first derivative of such function is not defined on the point x_0 where the gradient switches signs, but since m_a^- is convex the choice $d'_a(x_0) = 0$ is appropriate. Thus each component of the first derivative of M_f^- is given above. The lower bound of M_f^-, with arbitrary point $y_1 \leq x_0 \leq y_2$ is

$$LB(M_f^-) = \min\{l(y_1), l(y_2)\}, \text{ where } l(x) = (M_f^-(x_0))'(x - x_0) + M_f^-(x_0)$$

In the algorithms we have used the midpoint of the segment, $x_0 = \frac{y_1+y_2}{2}$. A DC upper bound of M_f utilizing the results above is given by

$$UB(M_f) = UB(M_f^+) - LB(M_f^-).$$

For the estimation of the lower bound on the operational cost we considered two very similar methods to the previous upper bound.

1. Using interval arithmetic, overestimating the distance to the demand points by assuming the follower delivers from the furthest point in its segment.
2. Taking advantage of the operational cost being DC and calculating a DC lower bound.

Let us assume that the function ψ in the operational cost is $\psi(t) = t^\mu + b$, where b is a positive constant. When overestimating the distance of the follower we underestimate the operational cost of the facility.

The DC decomposition of the cost function is very similar to the DC decomposition of the market share as both are S-shaped functions. Having calculated the required bound on both M_f and G_f we can define upper bounds on F_f by combining any of the two discussed bounds

$$UB(F_f) = UB(M_f) - LB(G_f)$$

3.3 Algorithm

We use a Branch and Bound method to find a solution that differs less than ϵ_f from the optimum. The algorithm's pseudo code is described in Algorithm 1, with lines 27 to 29 omitted as those apply only when the leader problem is solved. For the sake of simplicity let us denote the follower's objective function by F, and the sets of initial partitions by Λ_0. The initial partitions can vary, in principle the follower problem is to be solved on the whole network, thus Λ_0 contains every edge of the network, however if we already know that a set of segments contain the global optimum, it is enough that Λ_0 contain those segments. We refer to such set of segments as *partial solutions*, together with a given segment or point for the leader's facility.

The algorithm starts in line 1-3 with defining the initial set of partial solutions Λ, initializing the global lower bound z^L and the upper bound z_i^U for

every initial subproblem. The cycle from line 4 to line 11 computes proper upper bounds for the initial subproblems (line 5) as well as determines lower bounds z_t^L (lines 6-7) using the midpoint of the given segment. In line 8 we guarantee that only the best lower bound is kept as the global lower bound and the point on which this was achieved is stored in *BestPoint*. After the initial calculations we remove each segment from Λ known to not contain the global optimizer in line 12.

The cycle from line 14 to line 35 is the main cycle of the general Branch and Bound method for global optimization. The selection rule selects the subproblem with the highest upper bound. The branching rule used splits the segment of the partial solution along its midpoint.

4 Leader problem

In this section we describe the Branch and Bound method used to solve the leader problem. When we are talking about partial solutions or subproblems in this section we refer to a segment of the leader and the set of follower segments associated with it. The main goal of the method is for every subproblem simultaneously tighten the set containing the global optimizer of the leader and every follower segment that can contain the global optimizer for the follower problem.

4.1 Lower bound

Let us first consider that when estimating a lower bound for the leader, the follower's market share should be overestimated, thus underestimating the leader's profit function. We can use a similar approach like we used for the follower problem. The best objective value at a finite set of feasible solutions $\{x_l^1, x_l^2, \dots x_l^r\}$ is

$$z^L = \max\{F_l(x_l^1, x_f^{1*}), F_l(x_l^2, x_f^{2*}), \dots F_l(x_l^r, x_f^{r*})\}$$

We considered two options:

1. We could use the previous B&B method for finding the optimal solution x_f^{i*} to the follower problems.
2. Or we can use interval arithmetic and underestimate the follower's distance from each demand point in the objective function of the leader.

When using the follower Branch and Bound algorithm described in Subsection 3.3 one should use a different stopping criterion in this case. Either a sufficiently large ϵ_f should be considered, or better just running the algorithm for a fixed number of iterations. This is due to the fact that determining an adequate solution to the follower problem can take significant computational effort.

Underestimating the follower's distance is similar to the method of distance underestimation we used in Subsection 3.2, except that here we need to underestimate on multiple segments, due to the leader having multiple follower segments associated with it. We should underestimate the distance of each segment to a

demand point, and then use the shortest distance obtained, overestimating the market share of the follower and thus underestimating the profit of the leader. This is equivalent to using the natural inclusion function of F_l on the union of follower segments and the given x_l leader position.

4.2 Upper bound

To calculate an upper bound for the leader problem on a given segment we use a similar method we used for the lower bound except that we need to keep the leader as an interval. Again we considered two options:

1. We could use the follower B&B method modified to use an interval for the position of the leader.

2. Or we can use interval arithmetic overestimating the follower's distance and underestimating the leader's distance from each demand point.

Let us consider the modified B&B method for the follower problem. Its objective function would not converge to an optimal point but to an optimal interval, due to the leader's position being given as an interval. This makes the algorithm computationally more expensive, because each bound estimation has to be made on a two dimensional interval as opposed to a one dimensional one as in the original algorithm. After a fixed amount of iterations the method should yield the follower segment with the highest lower bound. The result can be used to calculate an upper bound using interval arithmetic.

For our second option that is using interval arithmetic, we can calculate the upper bound using the natural inclusion function of M_l on the union of follower segments and the leader segment.

In both cases we need to use a separate method to underestimate the operational cost of the leader. This can be done the same way as in Subsection 3.2, using DC decomposition or interval arithmetic.

4.3 Refining the follower segments

The key to this algorithm is to refine both the leader's partition and the follower partitions for each leader segment as well. For the convergence of the leader bounds we assumed a single optimizer point for the follower. We considered two methods to accomplish the refinement:

1. Using the modified follower B&B method described in the previous subsection for a fixed number of iterations.

2. Refining each follower segment to be at most as wide as the leader segment it is associated with.

When using the B&B method we must be careful not to eliminate the segment on which BestPoint is found, but apart from that, the algorithm is the same as before.

The other method is similar to Algorithm 1. We denote the follower's objective function by F. The algorithm modifies the input list of follower segments Λ, it is in essence a Branch and Bound method, with a stopping criterion of having each follower segment's diameter be less than the input width δ, thus the condition on line 14 of the algorithm needs to be changed to $Diam(C) \not\leq \delta \ \forall C \in \Lambda$. The output of the algorithm is the modified list of segments. The selection rule selects the widest segment, while the branching rule is the same as the follower's, splitting the given segment along its midpoint. In this case *BestPoint* does not need to be maintained through the algorithm.

4.4 The main algorithm

A similar B&B algorithm is used for the leader problem as for the follower. Let us denote now the leader's objective function by F. The method's pseudo-code is described in Algorithm 1, it is the practically the follower's B&B method with the refinement of the follower segments added in lines 27 to 29. In fact the additional differences of the follower's and leader's procedure are hidden in the bound calculations. The initial set Λ_0 is the edges of the network as leader segments, and for every leader segment associate every edge as a follower segment with it. The selection rule selects the partial solution with the highest upper bound, while the branching rule bisects the leader segment along its midpoint and leaves the follower segments unchanged, but duplicated for the new segments.

5 Computational results

In this chapter we present the computational results obtained by using the algorithm described in Subsection 4.4. The algorithm was written in C++ using the PROFIL/BIAS library [3], and executed on an Intel Xeon computer with 256GB memory at 2GHz. Throughout the execution in no cases has the memory requirement exceeded 2GB.

The accuracy used for both the leader and the follower problems was 10^{-3}. The attraction was chosen according to the gravitational model, i.e. $\varphi(t) = t^2$, while the function for the operational costs was $\psi(t) = t^2 + b$, where $b = 10$ was used. The containers for the partial solutions were augmented red-black trees in all cases, sorted according to the selection rule used. The algorithm used interval arithmetic upper bounds for the follower and leader calculations, to refine the follower segments we used the second method described in Subsection 4.3, where the follower segments are kept to be at most as wide as their associated leader segments. The other method that used the follower B&B to refine the follower segments proved to be unreliable, either refining the segments unnecessarily or not refining them enough. More research is needed to obtain suitable parameters for it to be useful.

Algorithm 1: General B&B method used

Data: Λ_0
$\Lambda := \Lambda_0$
$z^L := 0$
$z_i^U := \infty$ for all i
for $t := 1$ **to** $|\Lambda|$ **do**
  Determine an upper bound z_t^U of F over C_t
  $y_t := \text{midpoint}(C_t)$
  Compute lower bound $z_t^L := F(y_t)$
  if $z_t^L > z^L$ **then**
    $z^L := z_t^L$, $BestPoint := y_t$
  end
end
Remove all C_i from Λ with $z_i^U < z^L$
$r := 0$
while $\Lambda \neq \emptyset$ **do**
  Select C from Λ
  Bisect C into C_{r+1} and C_{r+2}
  for $t := r + 1$ **to** $r + 2$ **do**
    Determine an upper bound z_t^U on C_t
    if $z_t^U > z^L + \epsilon_f$ **then**
      Select $y_t := \text{midpoint}(C_t)$
      Compute $z_t := F(y_t)$
      if $z_t > z^L$ **then**
        $z^L := z_t$, $BestPoint := y_t$
        Remove all C_i from Λ with $z_i^U < z^L$
      end
      if $z_t^U > z^L + \epsilon_f$ **then**
        if *leader problem* **then**
          Refine the follower segments of C_t
        end
        $\Lambda := \Lambda \cup \{C_t\}$
      end
    end
  end
  $r := r + 2$
end
Result: $\{BestPoint, z^L\}$

5.1 Results on a single network

The algorithm was first tested on a 24-node 39-edge network named Sioux-Falls from [5], with several instances of the problem generated using different number of facilities and different distributions of existing facilities between the leader and the follower. The number of facilities tested were $m \in \{0, 4, 8, 16, 32\}$ each with distributions $\%k/m \in \{25\%, 50\%, 75\%\}$ denoting the percentage of facilities owned by the leader, obviously for zero existing facilities the distribution was omitted. For each $(m, \%k/m)$ pair 10 problems were generated, with different demands on demand points uniformly distributed in the interval $[0, 10]$, also with different placement of existing facilities. The location of the facilities was generated by first choosing an edge e randomly, then placing the facility uniformly on the interval $[0, l_e]$.

The results are displayed in Table 1. Column *List size* shows the maximal overall size of the partial solutions lists during execution, for every leader segment every associated follower segment is taken into account, thus in fact showing the maximal memory requirements. The columns *Time* show the CPU time in seconds. In each case the minimal and maximal values are shown, as well as the mean and standard deviation.

m	$\%k/m$	List size		Time	
		[Min,Max]	*Mean ±Std*	*[Min,Max]*	*Mean ±Std*
0		[257185, 605092]	340979 ±98226.0	[63.0, 181.6]	92.6 ±33.3
4	25%	[83503, 4726751]	240709 ±1361805.9	[25.2, 1370.6]	62.1 ±394.0
	50%	[116300, 16237467]	205658 ±4876224.4	[31.3, 3985.7]	58.7 ±1196.4
	75%	[172861, 4171472]	232896 ±1239699.0	[41.9, 1327.6]	62.7 ±396.9
8	25%	[37698, 357136]	108429 ± 81066.3	[16.2, 144.7]	41.9 ±34.2
	50%	[98250, 4686779]	179371 ±1399935.4	[25.7, 1262.3]	58.1 ±367.6
	75%	[39762, 11117836]	835870 ±4028087.2	[17.9, 2651.9]	260.7 ±974.0
16	25%	[25351, 111317]	72754 ± 22863.4	[7.8, 56.5]	23.1 ±12.2
	50%	[35166, 165962]	85466 ± 43415.1	[10.9, 66.2]	35.8 ±18.6
	75%	[39371, 9026629]	97675 ±2683815.5	[12.3, 1771.9]	36.0 ±522.8
32	25%	[22451, 124665]	62317 ± 34206.9	[5.8, 44.5]	19.2 ±13.0
	50%	[31184, 604847]	49419 ±167447.9	[10.4, 182.6]	19.2 ±49.4
	75%	[25073, 200182]	58029 ± 49634.1	[6.6, 59.9]	19.9 ±14.4

Table 1: Results of the test instances for the leader problem executed on a network from [5]

The first interesting observation about the results is that the problem seems to become easier when more existing facilities are present. This is likely due to the objective function of the leader having more local optima, as it has more options to gain similar market share for low number of existing facilities. This phenomena was found in [6] as well. The other reason could be that with high number of facilities, the placement of a new facility can not alter the market share as much as with low number of facilities. In that case, the

objective function of the leader is mostly defined by the operational cost, which is unaffected by the location of the follower, thus resulting in an easier problem.

By observing the minimum of the values as well as the means we can deduce that the problem becomes harder when more facilities owned by the leader. That is most likely caused by the leader having to counteract the follower's choice rather than to choose a profitable location, since at higher number of facilities the leader actually has to minimize its losses as seen in [6].

5.2 Results on randomly generated networks

In this section we aim to observe the computational effort of the algorithm as related to the number of nodes and edges in a network. The networks used in these test were created using the Watts and Strogatz model [7] to generate random small-world graphs with node numbers $|V| \in \{10, 15, \ldots, 50\}$, from which some edges of these graphs were trimmed to reach the desired number of edges while maintaining connectivity. Two graphs were generated for each $|V|$, one with $|E| = 1.3|V|$ and another with $|E| = 1.6|V|$. The assignment of demand values and facility placements were done as in the previous section, using uniform distribution and generating 10 facility placements for each network. The length of the edges were selected from the set $\{1, 2, \ldots, 10\}$ independently with replacement. The results are shown in Tables 2, 3, 4 with the same columns as before but omitting the minimum and maximum columns.

It is worth noting that, the algorithm handles problems with multiple optima or optima closer than the chosen ϵ very poorly. This is especially true when the follower's objective function has multiple optima. In that case despite the leader's segment gets small if the follower has two optimal points that give significantly different leader objective values, the leader's lower and upper bounds will never converge to a single value. What that means is that the condition on line 26 of Algorithm 1 will always be true for this subproblem in the leader's B&B, thus the while cycle's condition at line 14 will never fail, causing an endless loop. The case of a local optimum being very close to the global optimum isn't as severe, it will not cause an endless loop, but the execution time will heavily increase. This might explain the high standard deviation in the following results, and gives room for improvements.

The results show that the computational effort required increases with the number of nodes, as well as the number of edges. An interesting observation is that for the same number of nodes the higher number of edges $|E| = 1.6|V|$ results in lower standard deviations, showing that the algorithm becomes more stable with higher number of edges in a network. That might be due to lower probability of local optima being close to the global optimum, since the location space is larger, and thus the facilities are distributed with greater spacing leading to a steeper function for the market share.

Interestingly enough, the property that the lower the number of facilities the easier the problem, seems to only occur with $|V| \geq 25$, this can also be explained by the the location space being too small for 16, 32 facilities on smaller networks. The other property that being the higher the number of leader facilities the more

(|V|, |E|)	m	$\%k/m$	List size		Time		|V|, |E|	m	$\%k/m$	List size		Time	
			Mean	*Std*	*Mean*	*Std*				*Mean*	*Std*	*Mean*	*Std*
(10, 13)	0		59289	29274.9	7.4	2.5	(10, 16)	0		114949	49443.1	14.8	7.2
	4	25%	41790	2248672.5	5.9	635.4		4	25%	49979	46737.9	6.6	7.1
		50%	86569	924195.3	11.4	132.6			50%	96403	79843.5	12.7	10.9
		75%	198226	5410071.8	17.4	469.8			75%	76912	514401.9	14.0	64.5
	8	25%	20221	131244.4	3.3	41.1		8	25%	73749	49241.9	10.3	7.5
		50%	34105	137763.1	5.5	19.7			50%	40065	30914.6	6.5	5.0
		75%	82325	161704.1	11.9	20.0			75%	61036	27274.1	8.7	3.6
	16	25%	26233	74814.2	4.4	8.7		16	25%	99728	177092.8	12.2	19.6
		50%	17140	17940.1	2.2	1.9			50%	49523	38557.8	7.0	5.6
		75%	41179	109123.5	5.5	12.3			75%	74146	108927.6	7.8	13.7
	32	25%	26816	28751.6	3.2	4.4		32	25%	139590	112290.2	19.0	12.1
		50%	103360	738912.6	13.9	141.4			50%	54543	34189.2	8.4	4.2
		75%	23459	33197.4	3.3	2.8			75%	88285	508726.9	10.6	99.1
Average			116756	1400389.1	14.4	209.0	Average			153094	232829.0	20.2	34.3
(15, 19)	0		120117	19507.3	18.9	3.1	(15, 24)	0		164020	21460.4	25.9	7.2
	4	25%	105253	95907.0	24.4	22.5		4	25%	152659	3115985.1	24.1	504.9
		50%	390839	5359630.1	72.0	1505.8			50%	127014	50247.0	22.9	12.9
		75%	118911	11430956.2	23.5	1632.0			75%	291208	10861647.3	65.6	1831.4
	8	25%	52602	889839.6	13.2	199.4		8	25%	71385	58419.7	17.8	12.2
		50%	49106	39311.4	11.5	9.1			50%	132907	297991.6	22.6	55.9
		75%	65852	6827638.0	18.6	974.4			75%	104710	3338826.9	31.2	496.8
	16	25%	41975	51175.2	13.6	8.4		16	25%	74297	116126.0	18.5	18.0
		50%	61777	3486832.1	13.5	509.7			50%	89710	119753.1	21.0	19.3
		75%	70479	187199.3	21.2	28.9			75%	86637	196763.7	23.0	29.0
	32	25%	109109	67973.0	18.3	12.4		32	25%	99185	113128.8	21.3	18.5
		50%	99020	210523.4	20.8	31.0			50%	63768	66865.5	16.7	13.3
		75%	66680	966781.1	18.5	129.8			75%	77781	168929.4	16.2	36.0
Average			197110	3720906.1	40.5	636.8	Average			232801	2325387.5	47.1	383.8
(20, 26)	0		134731	36248.0	28.4	16.6	(20, 32)	0		304606	55781.6	54.7	13.2
	4	25%	200730	1158173.8	50.9	243.3		4	25%	159082	116302.3	49.6	27.4
		50%	157382	77249.9	32.7	13.9			50%	182231	107370.7	41.8	21.2
		75%	107643	121895.9	23.9	33.8			75%	116871	859771.0	26.9	188.0
	8	25%	140586	3596762.4	32.9	1074.2		8	25%	102785	2039710.7	26.2	544.8
		50%	111683	239362.1	41.6	50.1			50%	120046	2221639.0	34.0	377.0
		75%	55838	1044536.6	21.4	194.4			75%	161838	1506789.5	51.7	299.5
	16	25%	59914	229166.7	16.8	40.5		16	25%	50633	50082.2	13.0	18.8
		50%	59201	1569838.7	23.0	261.8			50%	104832	2767241.3	27.5	649.5
		75%	79663	8318588.4	20.7	1269.9			75%	81721	17003410.7	28.6	2842.6
	32	25%	210853	319688.4	57.6	91.9		32	25%	61554	464177.5	19.9	75.5
		50%	78427	3552577.9	21.3	489.7			50%	141828	91532.8	50.9	23.3
		75%	44264	125358.5	16.3	26.8			75%	101854	59426.6	34.9	18.1
Average			219536	2556427.8	58.1	478.7	Average			287819	3439115.6	73.5	643.1

Table 2: Results of test instances for the leader problem with nodes ranging from 10 to 20

(|V|, |E|)	m	$\%k/m$	List size		Time		|V|, |E|	m	$\%k/m$	List size		Time	
			Mean	*Std*	*Mean*	*Std*				*Mean*	*Std*	*Mean*	*Std*
(25, 32)	0		95232	14186.3	24.0	2.6	(25, 40)	0		199441	21403.1	48.0	5.9
	4	25%	144020	4758367.6	43.1	1550.5		4	25%	181006	50976.1	58.0	14.2
		50%	218849	487845.4	56.3	156.9			50%	195389	53551.3	57.2	9.3
		75%	141235	260002.4	46.3	103.6			75%	167961	76730.9	59.3	37.6
	8	25%	76560	38171.0	36.0	11.8		8	25%	126245	58032.0	42.8	21.6
		50%	510037	8463954.9	146.5	2579.1			50%	137417	38633.9	49.8	14.8
		75%	178792	2760055.6	65.7	657.5			75%	159640	2679830.0	56.0	702.6
	16	25%	59973	15019.1	29.0	9.5		16	25%	95639	48294.4	34.5	18.5
		50%	66662	628386.3	28.9	146.4			50%	111608	57368.6	52.8	20.8
		75%	114920	953855.5	42.6	246.6			75%	114049	62940.2	53.3	25.8
	32	25%	40506	85977.1	19.4	20.1		32	25%	98901	132108.6	34.4	32.1
		50%	43620	142630.1	16.8	37.6			50%	85329	72143.2	30.2	25.1
		75%	89879	208826.5	37.2	49.7			75%	203656	291313.3	69.7	74.9
Average			250062	2357544.6	81.3	698.4	Average			288953	463574.4	95.4	128.6
(30, 39)	0		183166	41115.1	55.2	11.1	(30, 48)	0		218876	48181.2	57.1	11.7
	4	25%	202144	159190.6	67.5	70.9		4	25%	284624	107567.2	121.0	61.2
		50%	257754	13135498.7	84.5	3280.9			50%	291444	624394.8	125.3	182.6
		75%	158087	381033.6	69.5	136.5			75%	192425	112871.9	72.9	66.5
	8	25%	244771	4792373.8	90.9	1390.7		8	25%	230773	715282.1	92.3	219.1
		50%	265347	4351914.9	76.2	1317.7			50%	191541	162129.2	60.5	79.4
		75%	461794	3189850.4	240.8	911.3			75%	471855	794828.8	180.9	252.0
	16	25%	95276	70402.6	44.6	27.6		16	25%	170128	1081362.9	76.5	397.0
		50%	106415	885066.0	51.0	259.1			50%	147367	301673.7	65.9	88.8
		75%	146980	614195.1	68.3	225.0			75%	227571	136146.9	88.3	66.4
	32	25%	53999	50078.8	23.7	18.4		32	25%	73278	37482.4	47.6	16.5
		50%	64570	23950.2	23.9	16.7			50%	112638	47440.3	45.1	24.6
		75%	239128	10668370.5	106.0	2739.0			75%	119702	1705995.3	62.5	413.7
Average			359030	4809849.3	140.2	1304.7	Average			398514	746594.3	153.2	238.2
(35, 45)	0		122759	21568.5	40.1	6.8	(35, 56)	0		298032	49014.8	109.1	15.1
	4	25%	229805	2817379.8	114.4	1176.3		4	25%	346755	361827.0	158.8	223.0
		50%	201878	51069.3	81.5	21.1			50%	389234	129445.1	191.1	69.3
		75%	213062	199214.5	91.7	129.0			75%	476757	2679414.7	226.4	1030.4
	8	25%	173661	303909.6	108.0	120.4		8	25%	295311	103978.8	139.1	58.6
		50%	205377	68164.5	80.7	42.4			50%	342536	5645778.2	162.7	2276.5
		75%	312676	5002146.4	163.4	1834.0			75%	341551	431501.4	181.6	211.8
	16	25%	69137	447826.1	34.5	116.3		16	25%	218299	58222.1	104.7	28.9
		50%	182141	529022.1	97.2	202.5			50%	274374	1420569.4	129.7	391.6
		75%	151541	1558351.8	89.0	473.0			75%	335604	944585.6	203.2	411.2
	32	25%	68004	450539.5	47.5	143.5		32	25%	116223	53937.0	60.0	22.5
		50%	92062	1154929.8	44.6	375.3			50%	157304	4940627.2	77.8	1626.2
		75%	242994	6146629.5	110.1	2268.1			75%	253399	3795093.3	146.3	1139.5
Average			315094	2348624.5	148.6	866.2	Average			556148	2591107.5	267.7	943.9

Table 3: Results of test instances for the leader problem with nodes ranging from 25 to 35

(|V|, |E|)	m	$\%k/m$	List size		Time		|V|, |E|	m	$\%k/m$	List size		Time	
			Mean	*Std*	*Mean*	*Std*				*Mean*	*Std*	*Mean*	*Std*
(40, 52)	0		268077	39660.3	107.1	16.9	(40, 64)	0		566037	78816.9	223.1	31.7
	4	25%	224197	114297.4	108.9	63.2		4	25%	312036	715796.5	151.9	317.6
		50%	251683	109319.4	156.4	96.3			50%	377953	69969.8	210.8	66.8
		75%	510811	344252.2	277.8	175.8			75%	360057	3797134.0	180.4	1194.6
	8	25%	342486	358077.9	179.1	159.3		8	25%	331140	2086943.2	142.0	931.3
		50%	295559	406917.3	150.2	222.3			50%	306819	153888.0	193.5	81.2
		75%	210400	8282749.3	159.1	3924.9			75%	536581	4085980.5	335.4	1400.5
	16	25%	190164	98077.1	83.3	51.5		16	25%	201647	99045.6	106.4	45.7
		50%	277646	2264817.2	135.5	1022.9			50%	327034	268497.7	157.5	124.5
		75%	276890	147804.6	173.9	86.5			75%	287362	272459.1	162.3	151.7
	32	25%	88852	283718.9	48.9	115.5		32	25%	117954	36876.3	82.7	19.3
		50%	123405	38406.5	62.9	26.7			50%	146024	117558.2	85.9	68.6
		75%	260840	149817.0	112.0	76.4			75%	218089	3484052.4	104.9	1270.7
Average			474957	1588316.5	248.0	759.6	Average			633856	1923736.4	312.3	719.6
(45, 58)	0		262112	49063.6	104.0	29.2	(45, 72)	0		236642	19531.5	100.5	9.0
	4	25%	273968	128374.5	136.6	118.2		4	25%	297674	87368.1	154.7	34.7
		50%	194361	155806.8	117.1	149.5			50%	256403	59979.1	141.5	27.5
		75%	144954	372029.9	95.0	176.5			75%	212391	44922.9	113.8	28.4
	8	25%	208606	402633.3	117.2	173.3		8	25%	331885	112715.0	193.1	56.5
		50%	234769	2043697.3	210.0	1157.5			50%	423998	132312.5	279.4	73.1
		75%	287447	125706.1	205.5	75.7			75%	292930	106341.3	164.2	114.0
	16	25%	142610	80744.3	91.1	51.9		16	25%	234451	109995.8	170.7	46.6
		50%	237341	709700.7	123.2	291.5			50%	476219	869711.0	301.7	364.1
		75%	234409	158895.4	180.6	80.4			75%	248792	3935000.2	171.3	1556.8
	32	25%	102426	27622.0	59.1	22.8		32	25%	154874	84452.5	98.8	65.6
		50%	132169	63559.1	78.5	38.6			50%	263806	1809773.3	184.2	785.6
		75%	220230	341911.8	118.5	161.6			75%	278796	92964.4	189.7	51.3
Average			398753	598164.6	228.5	326.9	Average			525160	937997.6	309.7	404.6
(50, 65)	0		275271	27374.5	148.4	32.0	(50, 80)	0		642562	105282.4	311.2	64.2
	4	25%	353478	55476.4	275.5	86.7		4	25%	565368	165162.8	281.6	72.3
		50%	407292	530871.4	211.1	308.7			50%	508381	139240.6	333.6	110.6
		75%	476199	3938940.2	492.7	1912.2			75%	527411	242319.9	303.6	147.1
	8	25%	301342	3260519.4	226.8	1356.5		8	25%	498885	205516.0	311.2	133.5
		50%	629776	4346781.6	400.3	2323.7			50%	447088	3498305.5	284.7	1664.6
		75%	349840	2584584.1	269.5	1312.7			75%	386703	5053314.9	284.1	2025.7
	16	25%	292602	694268.7	172.1	397.4		16	25%	262088	1296797.0	182.8	606.0
		50%	241924	898207.1	139.7	438.0			50%	415949	369354.9	229.8	238.4
		75%	585371	1117892.8	409.3	584.4			75%	507743	4285788.9	275.7	2132.5
	32	25%	117356	50983.8	97.1	65.4		32	25%	169827	91392.8	136.2	58.8
		50%	201677	761580.7	126.6	326.5			50%	318172	77619.5	216.8	55.3
		75%	178375	11398753.3	137.6	3219.1			75%	298690	115540.3	243.2	77.4
Average			620965	3718103.3	420.9	1556.9	Average			853865	1978227.4	501.3	936.0

Table 4: Results of test instances for the leader problem with nodes ranging from 40 to 50

difficult the problem also appears only for $|V| \geq 25$.

6 Concluding remarks

In this paper a Huff-like Stackelberg problem on networks with operational costs is described and modeled. A reliable algorithmic solution is proposed using the general Branch and Bound method for global optimization. For the calculation of the bounds we used interval arithmetic, as well as DC decomposition.

The problem of the follower is solved using a Branch and Bound method, with interval arithmetic and DC bounds, that is built in the B&B algorithm for the leader, where similar methods were used for the bound calculations. In addition we used a B&B based method to refine the subproblems of the follower.

Results for the Sioux-Falls network as well as randomly generated networks were shown. We concluded that the problem becomes harder when more facilities owned by the leader are present or there are less existing facilities. Also, the algorithm becomes more stable with higher number of edges in the network. A drawback of the algorithm is that it does not handle problems well with local optima close to the global optimum.

Future research will be mainly based on accelerating the algorithm and incorporation more diverse bound calculations. The problem with the qualities of the new facilities as variables may also be considered.

References

[1] Rafael Blanquero, Emilio Carrizosa, Amaya Nogales-Gómez, and Frank Plastria. Single-facility huff location problems on networks. *Annals of Operations Research*, pages 1–21, 2013.

[2] Tammy Drezner and Zvi Drezner. Facility location in anticipation of future competition. *Location Science*, 6(1–4):155–173, 1998.

[3] O. Knüppel. PROFIL/BIAS - a fast interval library. *Computing*, 1(53):277–287, 1993.

[4] Hande Küçükaydin, Necati Aras, and I Kuban Altınel. Competitive facility location problem with attractiveness adjustment of the follower: A bilevel programming model and its solution. *European Journal of Operational Research*, 208(3):206–220, 2011.

[5] Larry J LeBlanc, Edward K Morlok, and William P Pierskalla. An efficient approach to solving the road network equilibrium traffic assignment problem. *Transportation Research*, 9(5):309–318, 1975.

[6] M.E. Sáiz, E.M.T. Hendrix, J. Fernández, and B. Pelegrín. On a branch-and-bound approach for a Huff-like Stackelberg location problem. *OR Spectrum*, 31:679–705, 2009.

[7] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440–442, 1998.