Computers & Operations Research 40 (2013) 2062–2070

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Branch-and-bound algorithm for a competitive facility location problem

Vladimir Beresnev*

Sobolev Institute of Mathematics, pr. Akad. Koptyuga, 4, Novosibirsk 630090, Russia

ARTICLE INFO

Available online 1 March 2013

Keywords:
Bilevel programming problem
Pseudo-Boolean function
Upper bound
Branch-and-bound algorithm
Local search

ABSTRACT

We study a mathematical model generalizing the well-known facility location problem. In this model we consider two competing sides successively placing their facilities and aiming to "capture" consumers, in order to make maximal profit. We state the problem as a bilevel integer programming problem, regarding optimal noncooperative solutions as optimal solutions. We propose a branch-and-bound algorithm for finding the optimal noncooperative solution. While constructing the algorithm, we represent our problem as the problem of maximizing a pseudo-Boolean function. An important ingredient of the algorithm is a method for calculating an upper bound for the values of the pseudo-Boolean function on subsets of solutions. We present the results of a simulation demonstrating the computational capabilities of the proposed algorithm.

1. Introduction

We consider a competitive facility location problem generalizing the well-known facility location problem [1,2]. In the simple facility location problem we consider a set of consumes (clients) with a given demand for a single commodity and a set of sites where facilities can be located. We consider the given fixed costs of opening facilities and assume there is a known profits that are made by satisfying the demand of each consumer from each facility. The simple facility location problem is to open a subset of facilities in order to maximize total profit under the condition that all demand has to be satisfied. In contrast to the simply facility location problem, the model under study includes two competing sides, which successively open their facilities aiming to "capture" consumers and maximize profit.

We may regard decision-making by the competing sides as a Stackelberg game [3] and, following the terminology of this game, call the sides a Leader and a Follower. The Leader's problem in this game consists in choosing a set of open facilities which yields maximal profit in the conditions that the Follower, knowing the Leader's facility locations and considering the preferences of consumers, will open some facilities. The Follower's problem consist in opening facilities to captures some consumers aiming to maximize his own profit.

The resulting mathematical model amounts to a bilevel integer programming problem [4] including an upper-level problem (the Leader's problem) and a lower-level problem (the Follower's problem). Both problems are location problems with order [5]. Orderings are necessary for specifying the preferences of consumers and determining the rules for capturing consumers by the Leader or the Follower.

The available publications devoted to competitive location problems pay little attention to this model. An exception is [6], studying a model in which the upper- and lower-level problems are similar to the classical facility location problem. A model close to ours is the $(r|p)$-centroid problem of [7]. In that model two competing sides (a Leader and a Follower) also successively open their facilities. The Leader has to place p facilities knowing that the Follower will react by placing r facilities. Each consumers is served by the side which places the nearest facility. Leader and Follower aim to obtain the maximum total demand of the consumers. An important feature of this model is that the objective functions of the Leader and the Follower differ only by a sign.

Among the proposed approaches to solving this problem, we should note [8–10]. In [8] the $(r|p)$-centroid problem is represented as a (0, 1)-programming problem with exponentially large numbers of variables and constraints. The optimal solution to a relaxed problem, which includes only "significant" variables and constraints, yields an upper bound for the values of the objective function of the original problem and makes it possible to construct an feasible solution. The procedure of successive growth of the set of significant variables and constraints proposed in [9] makes it possible to successively improve the upper bound and the resulting approximate solution to the original problem. An algorithm was proposed in [10], based on the possibility of representation of the

* Tel.: +7 3833306389.
E-mail address: beresnev@math.nsc.ru

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.02.023

problem of $(r|p)$-centroid in the form of a minimax (0, 1)-programming problem and its reduction to a "standard" (0, 1)-programming problem. Note that the problem in this paper cannot be represented as a minimax (0, 1)-programming problem, and so it requires some new approaches to constructing algorithms for solutions.

In this paper we propose a branch-and-bound algorithm [2,11] for the competitive facility location problem, which enables us to find optimal solutions to the problem. We regard as optimal the so-called optimal noncooperative solutions. This is related to the fact that, owing to the possible nonuniqueness of the optimal solution to the lower-level problem, the concept of optimality for the competitive facility location problem has to be refined.

A function from the set of (0, 1)-vectors to the reals is called a pseudo-Boolean function [12]. While constructing the algorithm, we represent the problem under study as the problem of maximizing a pseudo-Boolean function in as many variables as there are possible facility locations of the Leader. The considered function is defined implicitly; to calculate its values, we have to find optimal solutions to two integer linear programming problems.

An important ingredient of the algorithm is a method we propose for calculating an upper bound for the values of the pseudo-Boolean function under consideration on subsets of solutions. The idea of the method for calculating an upper bound of the objective function in a competitive facility location problem was first presented in [13]. In this paper, some upper bound is constructed in the linear case of the problem in which the profit from the consumer service is the same for all facilities. The results of the numerical experiment demonstrating the accuracy of the proposed upper bound are given in [14]. In this paper we construct an algorithm for calculating an upper bound in the problem under consideration in a general form. Simultaneously with the calculation of an upper bound we determine an feasible solution of the subset under consideration which yields a lower bound for the optimal value of the objective function. To calculate the initial record solution, the algorithm we propose uses the local search procedure of [14] with a neighborhood of a particular form.

This paper consists of six sections. In Section 2 we state the competitive facility location problem as a bilevel integer programming problem and define optimal noncooperative solutions. Moreover, we show that we can represent the problem of seeking optimal noncooperative solution as a problem of maximizing some pseudo-Boolean function. Section 3 describes the general scheme of the branch-and-bound algorithm for the problem of maximizing a pseudo-Boolean function. As applies to the set of (0, 1)-vectors, we specify the method for determining subsets of solutions and the branching function. To determine subsets of solutions, we use the so-called partial (0, 1)-vectors with some entries fixed. In Section 4 we consider a method for calculating an upper bound for the values of the pseudo-Boolean function under consideration on the subsets of (0, 1)-vectors determined by partial solutions, while in Section 5 we present a local search algorithm for calculating a locally optimal solution used on the first step of the branch-and-bound algorithm as the initial record solution. Section 6 is devoted to a discussion of the results of simulation with the proposed branch-and-bound algorithm. As a test we use sample problems of competitive location on a network from the library [15].

2. The competitive facility location problem

2.1. The model

The competitive facility location problem, as we already noted, is a mathematical model arising in studying a more general situation than the case of the classical location problem. Here we also have a set of potential facilities (facility locations) and a set of consumers. However, in contrast to the classical facility location problem, there are two competing sides, which successively place (open) facilities aiming to capture consumers and achieve their goals, different in general. Moreover, we assume that, as in the case of location problem with order, each consumer has their own preferences, which enable us to rank the open facilities and determine which of the sides captures the consumer. A side captures a consumer if its facility is more preferable for this consumer than every facility opened by the other side.

We may regard decision-making by the competing sides in this facility location scenario as a Stackelberg game. Following the terminology of this game, we refer to the competing sides respectively as the Leader and the Follower; moreover, we represent decision-making solutions as consisting of the following two stages.

At the first stage the Leader decides to place his facilities, aiming to maximize his profit. Furthermore, he knows the objective function of the Follower, as well as the preferences of each consumer.

At the second stage the Follower, knowing the Leader's facility locations, opens his facilities aiming to maximize his profit. Furthermore, he knows the preferences of each consumer.

The Leader's problem in this game consists in determining the set of open facilities which maximizes his objective function provided that the Follower captures some consumers.

In order to write the Leader's problem formally, we introduce some notation and state necessary assumptions. In particular, we formalize the consumer capturing rule by the competing sides, as well as the rules according to which for each consumer captured by the Leader or the Follower a facility is assigned to serve that consumer.

As in the classical facility location problem, denote by $I=\{1,\dots,m\}$ the set of facilities (possible facility locations) and by $J=\{1,\dots,n\}$ the set of consumers.

Assume that facility $i\in I$ can be opened by both the Leader and the Follower. Therefore, for every $i\in I$ we assume given two quantities f_i and g_i equal to the fixed cost to open facility i respectively by the Leader and the Follower. If for some reason the Leader or the Follower cannot open facility i then we put $f_i=\infty$ or $g_i=\infty$.

For all $i\in I$ and $j\in J$ denote by p_{ij} the profit made at facility i by serving consumer j.

Assume that the choice among the open facilities for serving consumer $j\in J$ is made accounting for the preferences of consumer j. Assume that the preferences of consumer $j\in J$ are given as a linear order relation $\succ_j$ on I. For i, $k\in I$ the relation $i\succ_j k$ means that out of two open facilities i and k consumer $j\in J$ prefers facility i. The relation $i\succcurlyeq_j k$ means that either $i\succ_j k$ or $i=k$.

Take $I_0\subset I$. For every $j\in J$ denote by $i_j(I_0)$ the best facility from set I_0, i.e. the element $i_0\in I_0$ with $i_0\succcurlyeq_j i$ for every $i\in I_0$. If $I_0=\{i\in I|w_i=1\}$, where $w=(w_i)$ is a (0, 1)-vector, we also write $i_j(w)$ instead of $i_j(I_0)$.

We use the following rule to determine the side capturing consumer $j\in J$. Suppose that the unit components of a (0, 1)-vector $x=(x_i)$ for $i\in I$ indicate the facilities opened by the Leader, while the unit components of a (0, 1)-vector $z=(z_i)$ for $i\in I$ indicate the facilities opened by the Follower. Consumer $j\in J$ is captured by the Leader whenever $i_j(x)\succ_j i_j(z)$ and by the Follower whenever $i_j(z)\succ_j i_j(x)$.

If consumer $j\in J$ is captured by the Follower for instance then to choose the facility to serve this consumer the Follower can use the following two rules. According to the first rule, which we call the *strict choice* rule, it will be facility $i_j(z)$, while according to the second rule, which we call the *free choice* rule, it can be some facility i opened by the Follower satisfying $i \succ_j i_j(x)$. Below, while constructing mathematical models, we assume that the Leader uses the strict choice rule, while the Follower, the free choice rule.

Assume that the goals of the Leader and the Follower are to make maximal profit. The Leader's profit is combined from the profit at all facilities opened by the Leader. In turn, the profit at each open facility equals the total revenue received from the consumers served there minus the fixed cost to open this facility. Similarly, the Follower's profit is combined from the profit at all facilities he opened, while the profit at each open facility is equal to the total revenue received from consumers minus the fixed cost.

Introduce the following variables similar to the variables of the classical facility location problem:

the value of x_i is 1 if the Leader opens facility $i \in I$ and zero otherwise;
the value of x_{ij} is 1 if facility $i \in I$ opened by the Leader is assigned to serve consumer $j \in J$ and zero otherwise;
the value of z_i is 1 if the Follower opens facility $i \in I$ and zero otherwise;
the value of z_{ij} is 1 if facility $i \in I$ opened by the Follower is assigned to serve consumer $j \in J$ and zero otherwise.

Using these variables, we can formulate the Leader's problem in the Stackelberg game under consideration, which we call the *competitive facility location problem*, as the following bilevel integer programming problem:

$$\max_{(x_i),(x_{ij})} \left\{ -\sum_{i \in I} f_i x_i + \sum_{j \in J} \left(\sum_{i \in I} p_{ij} x_{ij} \right) \left(1 - \sum_{i \in I} \tilde{z}_{ij} \right) \right\}; \tag{1}$$

$$x_i + \sum_{k: i \succ_j k} x_{kj} \le 1, \quad i \in I, \quad j \in J; \tag{2}$$

$$x_i \ge x_{ij}, \quad i \in I, \quad j \in J; \tag{3}$$

$$x_i,\ x_{ij} \in \{0,1\}, \quad i \in I, \quad j \in J; \tag{4}$$

$$(\tilde{z}_i),\ (\tilde{z}_{ij}) \quad \text{is the optimal solution to the problem} \tag{5}$$

$$\max_{(z_i),(z_{ij})} \left\{ -\sum_{i \in I} g_i z_i + \sum_{i \in I} \sum_{j \in J} p_{ij} z_{ij} \right\}; \tag{6}$$

$$x_i + \sum_{k: i \succcurlyeq_j k} z_{kj} \le 1, \quad i \in I, \quad j \in J; \tag{7}$$

$$z_i \ge z_{ij}, \quad i \in I, \quad j \in J; \tag{8}$$

$$z_i,\ z_{ij} \in \{0,1\}, \quad i \in I, \quad j \in J. \tag{9}$$

As every bilevel mathematical programming problem, (1)–(9) includes the upper-level problem (1)–(4) and the lower-level problem (6)–(9). In the problem (1)–(9) we call the upper-level problem as Leader's problem and denote it by **L**, we call the lower-level problem as Follower's problem and denote by **F**.

The objective function (1) of problem **L** expresses, as we already noted, the profit made by the Leader accounting for the loss of some consumers captured by the Follower. Eqs. (2)–(4) are constraints of a location problem with order. Inequalities (2) implement the strict choice rule of a facility opened by the Leader for serving consumers. These constraints show that if $x_i = 1$ for some $i \in I$ then $x_{kj} = 0$ for every $k \in I$ such that $i \succ_j k$. The same inequalities guarantee that only one facility opened by the Leader can be chosen to serve each consumer. The objective function (6) of problem **F** expresses the profit made by the Follower. Inequalities (7) implement conditions for capturing consumers by the Follower given the facilities opened by the Leader. These constraints show that if $x_i = 1$ for some $i \in I$ then $z_{kj} = 0$ for every $k \in I$ such that $i = k$ or $i \succ_j k$. In particular, these constraints show that if a facility is opened by the Leader then the Follower cannot use it for serving consumers. The remaining constraints of problem **F** are constraints of the classical facility location problem.

For the problem (1)–(9) as a whole we use the notation (**L**,**F**). We assume that the objective function (1) of problem **F** is also the objective function of problem (**L**,**F**).

The initial data for problem (**L**,**F**) are the matrices $P = (p_{ij})$, $i \in I$, $j \in J$; $F = (f_i)$, $i \in I$; $G = (g_i)$, $i \in I$, and $R = (r_{ij})$, $i \in I$, $j \in J$. The matrix R defines the orders $\succ_j$ for $j \in J$ on I. The entries in column j of R are distinct, and for all $i,k \in I$ the inequality $r_{ij} < r_{kj}$ holds if and only if $i \succ_j k$. We assume that the entries of P, F, and G are nonnegative, while those of R take values from 1 to m.

2.2. Feasible and optimal solutions

Henceforth we denote an feasible solution $((x_i),(x_{ij}))$ to problem **L** by X and an feasible solution $((z_i),(z_{ij}))$ to problem **F** by Z. Refer to a pair $(X,\tilde{Z})$, where X is an feasible solution to problem **L** and $\tilde{Z}$ is an optimal solution to problem **F**, as an *feasible solution* to problem (**L**,**F**).

Consider an feasible solution $(X,\tilde{Z})$ to problem (**L**,**F**) with $X = ((x_i),(x_{ij}))$ and $\tilde{Z} = ((\tilde{z}_i),(\tilde{z}_{ij}))$. Without loss of generality, assume that if X is a nonzero solution to problem **L** then $\sum_{i \in I} x_{ij} = 1$ for every $j \in J$. As for the optimal solution $\tilde{Z}$ to problem **F**, assume that for every $j \in J$ if $i_j(\tilde{z}) \succ_j i_j(x)$, that is, if consumer j is captured by the Follower, then $\sum_{i \in I} \tilde{z}_{ij} = 1$. In the case that all p_{ij} for $i \in I$ are positive, this condition holds automatically since $\tilde{Z}$ is an optimal solution. However, if some p_{ij} with $i \in I$ vanishes then the indicated condition must be violated. However, in this case if we set the value of the corresponding variable $\tilde{z}_{ij}$ with $i \in I$ to 1 then we obtain an optimal solution with the required property.

Assume also that if X is the zero solution then the optimal solutions $\tilde{Z}$ to problem **F** are nonzero. Therefore, henceforth we assume that the feasible solutions $(X,\tilde{Z})$ to problem (**L**,**F**) are nonzero.

Denote by $L(X,\tilde{Z})$ the value of the objective function of problem (**L**,**F**) on an feasible solution $(X,\tilde{Z})$ and by $F(Z)$, the value of the objective function of problem **F** on an feasible solution Z.

Observe that problem (**L**,**F**) is well-posed if all feasible solutions $(X,\tilde{Z}_1)$ and $(X,\tilde{Z}_2)$ satisfy $L(X,\tilde{Z}_1) = L(X,\tilde{Z}_2)$. This condition holds, in particular, when problem **F** has a unique optimal solution for every X. However, if this condition is violated then problem (**L**,**F**) is not well-posed since for some X it is unclear which optimal solution $\tilde{Z}$ to problem **F** we should use for calculating the values of the objective function of problem (**L**,**F**).

In order to remove this indeterminacy, we have to accept additional rules which the Follower applies to choose his solution. Consider the following rule for the Follower to choose his solution, which we call the *noncooperative behavior* rule.

Assume that in noncooperative behavior the Follower chooses among the solutions optimizing his objective function the solution which is the worst from the viewpoint of the Leader and yields the least value of the Leader's objective function. Refer to an feasible solution $(X,\overline{Z})$ of problem (**L**,**F**) as an *feasible noncooperative solution* to problem (**L**,**F**) if $L(X,\overline{Z}) \le L(X,\tilde{Z})$ for every feasible solution $(X,\tilde{Z})$ to problem (**L**,**F**). Refer to an feasible noncooperative solution $(X^*,\overline{Z}^*)$ to problem (**L**,**F**) as an *optimal noncooperative solution* to problem (**L**,**F**) if $L(X^*,\overline{Z}^*) \ge L(X,\overline{Z})$ for every feasible noncooperative solution $(X,\overline{Z})$.

Observe that every feasible noncooperative solution $(X,\overline{Z})$ to problem (**L**,**F**) is determined by the feasible solution X. For every feasible solution X the value of the objective function $L(X,\overline{Z})$ on the corresponding feasible noncooperative solution $(X,\overline{Z})$ is uniquely determined.

Indeed, consider an feasible solution $X=((x_i),(x_{ij}))$ to problem **L**. The corresponding feasible noncooperative solution $(X,\overline{Z})$ to problem (**L**,**F**) is determined by the algorithm consisting of two stages.

At stage 1 for a fixed solution X we solve problem **F** and calculate the optimal value F^* of its objective function.

At stage 2 for a fixed solution X we solve the auxiliary problem

$$\min_{(z_i),(z_{ij})} \sum_{j\in J}\left(\sum_{i\in I} p_{ij}x_{ij}\right)\left(1-\sum_{i\in I} z_{ij}\right); \tag{10}$$

$$x_i+\sum_{k:i\succ_j k} z_{kj}\le 1,\quad i\in I,\quad j\in J; \tag{11}$$

$$z_i\ge z_{ij},\quad i\in I,\quad j\in J; \tag{12}$$

$$-\sum_{i\in I} g_i z_i+\sum_{i\in I}\sum_{j\in J} p_{ij}z_{ij}\ge F^*; \tag{13}$$

$$z_i,z_{ij}\in\{0,1\},\quad i\in I,\quad j\in J. \tag{14}$$

The optimal solution $\overline{Z}=((\overline{z}_i),(\overline{z}_{ij}))$ to this problem yields the required feasible noncooperative solution $(X,\overline{Z})$ to problem (**L**,**F**). Furthermore, observe that $L(X,\overline{Z})$ is the same for every optimal solution $\overline{Z}$ to the auxiliary problem (10)–(14).

Henceforth we focus our attention on the model (**L**,**F**) and the problem of seeking optimal noncooperative solutions.

We saw above that every feasible solution X to problem **L** uniquely determines the value of the objective function of problem (**L**,**F**).

Observe also that the feasible solution $X=((x_i),(x_{ij}))$ to problem **L** is itself uniquely determined by a (0, 1)-vector $x=(x_i)$. Therefore, every (0, 1)-vector x uniquely determines some value $L(X,\overline{Z})$ of the objective function of the problem on the corresponding feasible noncooperative solution $(X,\overline{Z})$.

Thus, we can represent the problem of seeking the optimal noncooperative solution to the competitive facility location problem (**L**,**F**) as the problem of maximizing some pseudo-Boolean function $f(x)$, $x\in B^m$. This function is defined implicitly and in order to calculate its values we have to find the optimal solution to the lower-level problem (6)–(9) and then the optimal solution to the auxiliary problem (10)–(14). Therefore, while constructing below algorithms for the competitive facility location problem, we consider the problem of maximizing a pseudo-Boolean function $f(x)$ defined on the set of (0, 1)-vectors $x=(x_i)$ for $i\in I$ and construct algorithms for this problem.

2.3. The list of notation

Finally, we present a list of notations for convenience of further reading.

- $I=\{1,\dots,m\}$ the set of facilities
- $J=\{1,\dots,n\}$ the set of consumers
- f_i the fixed cost to open facility i by the Leader
- g_i the fixed cost to open facility i by the Follower
- p_{ij} the profit of facility i by serving consumer j
- $i\succ_j k$ the relation means that out of two open facilities i and k consumer j prefers facility i
- $i_j(I_0)$ the best facility for consumer j from the set of facility I_0
- $i_j(w)$ the best facility for consumer j from the set of facility $I_0=\{i\in I\,|\,w_i=1\}$, where $w=(w_i)$ is (0, 1)-vector
- x_i the (0, 1)-variables, $x_i=1$ if the Leader opens facility i and $x_i=0$ otherwise
- x_{ij} the (0, 1)-variables, $x_{ij}=1$ if facility i opened by the Leader is assigned to serve consumer j and $x_{ij}=0$ otherwise
- z_i the (0, 1)-variables, $z_i=1$ if the Follower opens facility i and $z_i=0$ otherwise
- z_{ij} the (0, 1)-variables, $z_{ij}=1$ if facility i opened by the Follower is assigned to serve consumer j and $z_{ij}=0$ otherwise
- **L** the Leader's problem
- **F** the Follower's problem
- (**L**,**F**) stated the Competitive Facility Location Problem
- $X=((x_i),(x_{ij}))$ feasible solution to problem **L**
- $Z=((z_i),(z_{ij}))$ feasible solution to problem **F**
- $\tilde{Z}=((\tilde{z}_i),(\tilde{z}_{ij}))$ optimal solution to problem **F**
- $(X,\tilde{Z})$ feasible solution to problem (**L**,**F**)
- $L(X,\tilde{Z})$ the value of the objective function of problem (**L**,**F**)
- $(X,\overline{Z})$ feasible noncooperative solution to problem (**L**,**F**)

3. Branch-and-bound method

3.1. The general scheme of the branch-and-bound algorithm

Consider the general scheme of the branch-and-bound method with depth-first search for the problem of maximizing a pseudo-Boolean function $f(x)$; thus, as applies to a problem of the form

$$\max_x \quad f(x);$$
$$x\in B^m.$$

Denote the optimal solution to this problem by x^*.

Assume that we are given the following functions defined on the subsets $s\subset B^m$. Refer to a function $b(s)$ which determines an improper subset of the set s as a *branching function*. Refer to a function $H(s)$ satisfying $H(s)\ge f(x)$ for every $x\in s$ as an *upper bound*. Assume that $H(s)=f(x)$ whenever $s=\{x\}$. Consider also a function $x(s)$ which determines some solution in s.

An algorithm implementing the branch-and-bound method with depth-first search consists of finitely many similar steps. On each step we consider a set $D\subset B^m$ called the *set of candidate solutions*, a subset $d\subset D$ called the *set of examined solutions*, as well as a solution $x^0\in B^m$ which is the best one available up to this step, called the *record solution*.

On the first step we have $D=B^m$ and $d=B^m$, while x^0 is an arbitrary element of B^m.

Suppose that before the next step we have a set D of candidate solutions, a set $d\subset D$ of examined solutions, and a record solution x^0. The step starts with calculating $H(d)$ and $x(d)$. If $f(x(d))>f(x^0)$ then we put $x^0=x(d)$. The step consists in trying to find out whether d contains a solution better than the record solution. To this end, we test the validity of

$$H(d)\le f(x^0).$$

If the inequality holds then d is discarded and we put $D=D\backslash d$. If $D=\emptyset$ then the algorithm stops; otherwise, we put $d=b(D)$ and start the next step. However, if the inequality is violated then we put $d=b(d)$ and start the next step.

Since on each step of the algorithm the examined subset is either discarded or replaced with an improper subset and since every examined singleton subset is discarded then the algorithm stops after a finite number of steps. The solution x^0 produced satisfies

$$f(x^*)\le f(x^0).$$

Indeed, if $x^0=x^*$ then the inequality holds. Suppose that $x^0\ne x^*$. Then on some step of the algorithm the solution x^* was discarded together with a set d. However, then the following inequalities hold, which prove the claim:

$$f(x^*) \le H(d) \le f(x^0).$$

In order to make this computational branch-and-bound scheme into a concrete algorithm for maximizing a pseudo-Boolean function, we have to fill the details of all ingredients of this scheme:

the method for specifying subsets D and d;
the branching function $b(s)$;
the method for calculating the upper bound $H(s)$;
the method for constructing the solution $x(s)$.

3.2. Specifying subsets of the set of (0, 1)-vectors

In the case of the set B^m it is convenient to define subsets of solutions using the so-called partial solutions. Put $I=\{1,\ldots,m\}$. Refer to a vector $y=(y_i)$ for $i\in I$ whose components take values 0, 1, and $*$ as a *partial* (0, 1)-*vector* or a *partial solution*. A partial solution divides the variables of the problem of maximizing a pseudo-Boolean function into variables with a specified value 0 or 1 and free variables. Given a partial (0, 1)-vector $y=(y_i)$ for $i\in I$, put

$$I^0(y)=\{i\in I \mid y_i=0\}, \quad I^1(y)=\{i\in I \mid y_i=1\}.$$

Refer to a vector $x\in B^m$ as an *extension* of a partial solution y whenever $I^0(y)\subset I^0(x)$ and $I^1(y)\subset I^1(x)$. Denote by $P(y)$ the set of all extensions of a partial solution y.

Call a partial solution y *ordered* whenever an ordering vector $(i_1,\ldots,i_q)$ with $\{i_1,\ldots,i_q\}=I^0(y)\cup I^1(y)$ is given for it, which indicates the order in which the components of the partial solution were assigned the values 0 or 1.

Verify that we can define the branching function so that on each step of the branch-and-bound algorithm the set D of candidate solutions and the set d of examined solutions are determined by some ordered partial solution.

Given an ordered partial solution y with ordering vector $(i_1,\ldots,i_q)$ for $1\le q\le m$, associate to it, apart from the set $P(y)$, another set $Q(y)$ with $P(y)\subset Q(y)$. In order to define $Q(y)$, for every k with $1\le k\le q$ such that $y_{i_k}=1$ construct a partial solution $y(k)=(y_i(k))$ for $i\in I$ satisfying

$$I^0(y(k))=(I^0(y)\cap\{i_1,\ldots,i_{k-1}\})\cup\{i_k\},$$
$$I^1(y(k))=I^1(y)\cap\{i_1,\ldots,i_{k-1}\}.$$

The union of the sets $P(y(k))$ for the constructed partial solutions $y(k)$ and the set $P(y)$ constitutes $Q(y)$.

Suppose that on some step of the branch-and-bound algorithm we have an ordered partial solution y with ordering vector $(i_1,\ldots,i_q)$, $1\le q\le m$, and suppose that D coincides with $Q(y)$, while d coincides with $P(y)$. Assume that on the first step we have a partial solution y satisfying $I^0(y)=\emptyset$ and $I^1(y)=\emptyset$, and then $P(y)=Q(y)=B^m$. Define the branching function so that on the next step of the algorithm the sets D and d are determined by some ordered partial solution.

Suppose that we apply the branching function to d. In this case $q<m$; therefore, we choose an element $i_{k+1}\notin I^0(y)\cup I^1(y)$ and consider a partial solution $y'=(y'_i)$ which differs from the initial partial solution y only in $y'_{i_{q+1}}=1$. The function $i(y)$ for choosing i_{q+1} has to be specified. It is not difficult to see that on the next step of the algorithm the set D and the subset d are determined by the ordered partial solution y' with ordering vector $(i_1,\ldots,i_q,i_{q+1})$. Suppose now that d is discarded on this step of the algorithm and we apply the branching function to $D\backslash d$. Denote by k_0 the largest index k, $1\le k\le q$, with $y_{i_k}=1$. If this index fails to exist then $D\backslash d\ne\emptyset$ and the algorithm stops. Otherwise, consider the partial solution $y'=(y'_i)$, which differs from the initial partial solution y in that $y'_{i_{k_0}}=0$ and $y'_{i_k}=*$ for $k=k_0+1,\ldots,q$. Then on the next step of the algorithm the set D and the subset d are determined by the ordered partial solution y' with ordering vector $(i_1,\ldots,i_{k_0})$.

4. An upper bound for the values of the objective function of the competitive facility location problem on subsets of solutions

Consider a method for calculating an upper bound $H(y)$ for the pseudo-Boolean function $f(x)$ under study on the set $P(y)$ of extensions of a partial solution $y=(y_i)$. The value $f(x)$ on a solution $x\in P(y)$ equals the value of the objective function of problem $(\mathbf{L},\mathbf{F})$ on the corresponding feasible noncooperative solution $(X,\overline{Z})$ with the additional property $x_i=y_i$ for $i\in I^0(y)\cup I^1(y)$. The maximal value of the pseudo-Boolean function $f(x)$ provided that $x\in P(y)$ equals the value of the objective function on the optimal non-cooperative solution to the following problem:

$$\max_{(x_i),(x_{ij})}\left\{-\sum_{i\in I}f_ix_i+\sum_{j\in J}\left(\sum_{i\in I}p_{ij}x_{ij}\right)\left(1-\sum_{i\in I}\tilde{z}_{ij}\right)\right\}; \tag{15}$$

$$x_i+\sum_{k:i\succ_j k}x_{kj}\le 1,\quad i\in I,\quad j\in J; \tag{16}$$

$$x_i\ge x_{ij},\quad i\in I,\quad j\in J; \tag{17}$$

$$x_i,x_{ij}\in\{0,1\},\quad i\in I,\quad j\in J; \tag{18}$$

$$x_i=y_i,\quad i\in I^0(y)\cup I^1(y); \tag{19}$$

$$(\tilde{z}_i),(\tilde{z}_{ij})\quad \text{is the optimal solution to the problem (6)–(9).} \tag{20}$$

Denote the problem (15)–(20) by $\mathbf{L}(y)$ and the problem (15)–(20), (6)–(9) by $(\mathbf{L}(y),\mathbf{F})$. An upper bound for the values of the objective function of problem $(\mathbf{L}(y),\mathbf{F})$ on the feasible noncooperative solutions $(X,\overline{Z})$ is the required upper bound for the values of the pseudo-Boolean function $f(x)$ on the set $P(y)$.

The method we propose for calculating an upper bound $H(y)$ rests on the construction of a system of subsets $\{I_j\}$, with $I_j\subset I$ for $j\in J$, using which we formulate a sufficient condition for the capture of consumers by the Follower.

Given a partial solution $y=(y_i)$, for a fixed element $j_0\in J$ we state conditions enabling us to find out for every $i\in I$ whether $i\in I_{j_0}$ or $i\notin I_{j_0}$.

If $y_i=0$ then $i\notin I_{j_0}$. Suppose that $y_i\ne 0$. Consider the set $N(i)=\{k\in I \mid k\succ_{j_0} i\}$. If $N(i)=\emptyset$ then $i\in I_{j_0}$. Suppose that $N(i)\ne\emptyset$. If $N(i)\cap I^1(y)\ne\emptyset$ then $i\notin I_{j_0}$.

Suppose that $N(i)\ne\emptyset$ and $N(i)\cap I^1(y)=\emptyset$. Consider the set

$$J(i)=\{j\in J \mid \text{if}\quad k\succ_j i_j(I^1(y)\cup\{i\})\quad \text{then } k\in N(i)\}.$$

Observe that $J(i)\ne\emptyset$ since $j_0\in J(i)$.

For every $k\in N(i)$ consider the set

$$J(k,i)=\{j\in J(i) \mid k\succ_j i_j(I^1(y)\cup\{i\})\}.$$

Assume that $i\in I_j$ whenever

$$g_k>\sum_{j\in J(k,i)}p_{kj}$$

for each $k\in N(i)$, and that $i\notin I_j$ if we find $k\in N(i)$ for which the inequality is violated.

The constructed subsets I_j for $j\in J$ are such that for every $j\in J$ the set $I_j\cap I^1(y)$ is either empty or coincides with $\{i_j(y)\}$.

We explain the meaning of the set I_j for $j \in J$ in the following lemma, which establishes that if the Leader plans on making a profit from consumer j using facility $i \notin I_j$ then consumer j is captured by the Follower.

Lemma 1. *Given a partial solution y and an feasible noncooperative solution* $(X,\overline{Z})$ *to problem* $(\mathbf{L}(y),\mathbf{F})$, *with* $X=((x_i),(x_{ij}))$ *and* $\overline{Z}=((\overline{z}_i),(\overline{z}_{ij}))$, *the equality* $\sum_{i\in I}\overline{z}_{ij_0}=1$ *holds for every* $j_0 \in J$ *such that* $p_{i_0j_0}x_{i_0j_0}>0$ *for some* $i_0 \notin I_{j_0}$.

Proof. Given (0, 1)-vectors $x=(x_i)$ and $\overline{z}=(\overline{z}_i)$, consider the elements $i_j=i_j(I^1(x)\cup I^1(\overline{z}))$. Suppose that $p_{i_0j_0}x_{i_0j_0}>0$ for some $j_0 \in J$ and $i_0 \notin I_{j_0}$, but the required equality is violated. Consider the set $N(i_0)=\{k\in I \mid k \succ_{j_0} i_0\}$ and observe that $N(i_0)\neq\emptyset$, while $x_i=0$ and $\overline{z}_i=0$ for $i\in N(i_0)$. Consider also the set $J(i_0)$ and observe that $i_j=i_j(I^1(y)\cup\{i_0\})$ for every $j\in J(i_0)$. Since $i_0\notin I_{j_0}$, it follows that we can find $k\in N(i_0)$ for which there exists a set $J(k,i_0)\subset J(i_0)$ such that $g_k \le \sum_{j\in J(k,i_0)} p_{kj}$. For this $k\in N(i_0)$ consider the set

$$S=\{j\notin J(i_0) \mid k\succ_j i_j, x_{i_j}=1\}$$

and construct a solution $Z=((z_i),(z_{ij}))$ to problem **F** which differs from of the optimal solution $\overline{Z}$ in that $z_k=1$ and $z_{kj}=1$ for $j\in J(k,i_0)\cup S$. The difference of the values of the objective function of problem **F** on $\overline{Z}$ and Z satisfies

$$F(Z)-F(\overline{Z})=-g_k+\sum_{j\in J(k,i_0)} p_{kj}+\sum_{j\in S} p_{kj}\ge 0.$$

This implies that Z is an optimal solution to problem **F**, while (X,Z) is an feasible solution to problem $(\mathbf{L}(y),\mathbf{F})$. The feasible noncooperative solution $(X,\overline{Z})$ and the feasible solution (X,Z) satisfy

$$L(X,\overline{Z})-L(X,Z)=\sum_{j\in J(k,i_0)}\sum_{i\in I} p_{ij}x_{ij}+\sum_{j\in S}\sum_{i\in I} p_{ij}x_{ij}\ge p_{i_0j_0}x_{i_0j_0}>0.$$

This contradicts the fact that $(X,\overline{Z})$ is an feasible noncooperative solution. The proof of the lemma is complete. □

Lemma 2. *Given a partial solution y and an feasible noncooperative solution* $(X,\overline{Z})$ *to problem* $(\mathbf{L}(y),\mathbf{F})$, *with* $X=((x_i),(x_{ij}))$ *and* $\overline{Z}=((\overline{z}_i),(\overline{z}_{ij}))$, *the equality*

$$\left(\sum_{i\in I} p_{ij}x_{ij}\right)\left(1-\sum_{i\in I}\overline{z}_{ij}\right)=\left(\sum_{i\in I_j} p_{ij}x_{ij}\right)\left(1-\sum_{i\in I}\overline{z}_{ij}\right)$$

holds for every $j\in J$.

Proof. If $p_{ij}x_{ij}=0$ for every $i\in I$ then the equality holds. Suppose that $p_{i_0j}x_{i_0j}>0$ for some $i\in I$. If $i_0\in I_j$ then the equality holds as well. However, if $i_0\notin I_j$ then the equality holds, since Lemma 1 yields $\sum_{i\in I}\overline{z}_{ij}=1$. The proof of the lemma is complete. □

For a fixed partial solution $y=(y_i)$ consider the following problem, which we call *estimational*:

$$\max_{(x_i),(x_{ij})}\left\{-\sum_{i\in I} f_i x_i+\sum_{j\in J}\sum_{i\in I_j} p_{ij}x_{ij}\right\},$$

$$x_i+\sum_{i:i\succ_j k} x_{kj}\le 1,\quad i\in I,\quad j\in J;$$

$$x_i\ge x_{ij},\quad i\in I,\quad j\in J;$$

$$x_i,x_{ij}\in\{0,1\},\quad i\in I,\quad j\in J;$$

$$x_i=y_i,\quad i\in I^0(y)\cup I^1(y).$$

Denote by $B(X)$ the value of the objective function of this problem on a solution $X=((x_i),(x_{ij}))$, while by $X^0=((x_i^0),(x_{ij}^0))$, the optimal solution to the problem.

Theorem 1. *Given a partial solution y, for every feasible noncooperative solution* $(X,\overline{Z})$ *to problem* $(\mathbf{L}(y),\mathbf{F})$, *the inequality* $L(X,\overline{Z})\le B(X^0)$ *holds.*

Proof. For every feasible noncooperative solution $(X,\overline{Z})$ to problem $(\mathbf{L}(y),\mathbf{F})$ Lemma 2 yields

$$L(X,\overline{Z})\le -\sum_{i\in I} f_i x_i+\sum_{j\in J}\sum_{i\in I_j} p_{ij}x_{ij}.$$

Since X is an feasible solution to the estimational problem, it follows that $L(X,\overline{Z})\le B(x)\le B(X^0)$. The proof of the theorem is complete. □

Considering the above, for a partial solution y define the values $H(y)$ and $x(y)$ as follows. If $y\notin B^m$ then consider the optimal solution $X^0=((x_i^0),(x_{ij}^0))$ to the estimational problem and put $H(y)=B(X^0)$ and $x(y)=x^0$, where $x^0=(x_i^0)$. If $y\in B^m$ then consider the feasible noncooperative solution $(X,\overline{Z})$ to problem $(\mathbf{L}(y),\mathbf{F})$ corresponding to the (0, 1)-vector y. Put $H(y)=L(X,\overline{Z})$ and $x(y)=y$.

Let us state, as a corollary to Theorem 1, a sufficient condition for the upper bound $H(y)$ calculated on a partial solution $y\notin B^m$ to be sharp and for $(X^0,\overline{Z})$ to be the optimal noncooperative solution to problem $(\mathbf{L}(y),\mathbf{F})$.

To this end, observe that

$$H(y)-L(X^0,\overline{Z})=B(X^0)-L(X^0,\overline{Z})=\sum_{j\in J}\left(\sum_{i\in I_j} p_{ij}x_{ij}\right)\sum_{i\in I}\overline{z}_{ij}.$$

This yields

Corollary 1. *For every partial solution* $y\notin B^m$ *the feasible noncooperative solution* $(X^*,\overline{Z})$ *is the optimal noncooperative solution to problem* $(\mathbf{L}(y),\mathbf{F})$ *provided that*

$$\left(\sum_{i\in I_j} p_{ij}x_{ij}^0\right)\sum_{i\in I}\overline{z}_{ij}=0$$

for every $j\in J$.

This implies in particular that if $\overline{Z}$ is the zero solution then the bound $H(y)$ is sharp and $(X^0,\overline{Z})$ is the optimal noncooperative solution to problem $(\mathbf{L}(y),\mathbf{F})$, while the (0, 1)-vector x^0 is the best solution in the set $P(y)$.

Using this criterion for the sharpness of the upper bound, accept the rule for choosing the potential element $i(y)$ for the branching function with the goal of the largest decrease on the next step of the difference $H(y)-f(x^0)$ between the upper and lower bounds for the maximal values of $f(x)$ on the corresponding subset of solutions.

Suppose that the set

$$J_0=\left\{j\in J \,\middle|\, \left(\sum_{i\in I_j} p_{ij}x_{ij}^0\right)\sum_{i\in I}\overline{z}_{ij}\neq 0\right\}$$

is not empty. For every $i\notin I^0(y)\cup I^1(y)$ consider the set

$$J_0(i)=\{j\in J_0 \mid i\succ_j i_j(\overline{z})\},$$

where $\overline{z}=(\overline{z}_i)$. Choose $i\notin I^0(y)\cup I^1(y)$ with the largest value of $\sum_{j\in J_0(i)} p_{ij}$ as $i(y)$.

5. Algorithm for calculating the initial record solution

On the first step of the branch-and-bound algorithm, when the partial solution y has no fixed components, we can take as the initial record solution the (0, 1)-vector $x(y)=x^0$ obtained as the solution to the corresponding estimational problem. However, since the initial record is important for speeding up the algorithm, consider an algorithm for improving the solution x^0 in order to obtain an initial record solution with a better value of the objective function. As a basis for this algorithm, considering the implicit definition of the function $f(x)$, we use the local search method, and in particular its simplest version: the standard local search algorithm.

The local search method is based on the concept of a neighborhood. As applies to the problem of maximizing a pseudo-Boolean function $f(x)$, we refer as a neighborhood of a point $x \in B^m$ to a subset $N(x) \subset B^m$ and assume that $N(x)$ is given for every $x \in B^m$. A solution x_0 with $f(x_0) \geq f(x)$ for every $x \in N(x_0)$ is called locally optimal.

The standard local search algorithm with a neighborhood $N(x)$ of $x \in B^m$ for the problem of maximizing a pseudo-Boolean function $f(x)$ consists of finitely many similar steps, on each of which we consider a current solution x_0. On the first step $x_0 = x^0$. The step consists in finding $x' \in N(x_0)$ maximally improving the current solution x_0; thus, an element $x' \in N(x_0)$ such that $f(x') \geq f(x)$ for every $x \in N(x_0)$ and $f(x') > f(x_0)$. If we cannot find a solution x' then the algorithm stops and the current solution x_0 is the required locally optimal solution. Otherwise, we replace the solution x_0 with x' and start the next step.

To implement this algorithm using a large neighborhood $N(x)$ can be quite difficult in view of the particular features of the pseudo-Boolean function $f(x)$ under consideration. Therefore, for a (0, 1)-vector $x = (x_i)$ define a neighborhood

$$N_0(x) \subset \{y \in B^m \,|\, d(x,y) \leq 2, |d(0,x) - d(0,y)| \leq 1\},$$

where $d(x,y)$ is the Hamming distance, which includes precisely m potential variants of modifications to x. To this end, for every $i \in I$ with $x_i = 1$ define

$$\Delta_i(x) = -f_i + \sum_{j : i = i_j(x)} p_{ij},$$

which we call the *profitability relative to solution* x of facility i opened by the Leader.

The neighborhood $N_0(x)$ will consist of (0, 1)-vectors $x^k = (x_i^k)$ for $k \in I$. For fixed $k \in I$ we construct x^k as follows.

If $x_k = 1$ then put $x_i^k = x_i$ for $i \neq k$ and $x_k^k = 0$. After that, assume that x^k is constructed.

If $x_k = 0$ then put $x_i^k = x_i$ for $i \neq k$ and $x_k^k = 1$. Then calculate $\Delta_k(x^k)$. Two cases are possible: $\Delta_k(x^k) \geq 0$ and $\Delta_k(x^k) < 0$. In the first case, when the profitability of facility k is nonnegative, for every $i \in I^1(x)$ calculate $\Delta_i(x^k)$. If $\Delta_i(x^k) \geq 0$ for every $i \in I^1(x)$, and so the opening of "new" facility k does not make some "old" facilities opened by the Leader unprofitable, then assume that x^k is constructed. However, if $\Delta_i(x^k) < 0$ for some $i \in I^1(x)$ then find $i_0 \in I^1(x)$ with $\Delta_{i_0}(x^k) \leq \Delta_i(x^k)$ for every $i \in I^1(x)$, put $x_{i_0}^k = 0$, and assume that x^k is constructed.

If $\Delta_k(x^k) < 0$, and so the profitability of the new facility is negative, then among the old facilities we find one whose removal maximally increases the profitability of the new facility k. To this end, for every $l \in I^1(x)$ construct the (0, 1)-vector $x^{kl} = (x_i^{kl})$, where $x_i^{kl} = x_i^k$ for $i \neq l$ and $x_l^{kl} = 0$, and calculate $\Delta_k(x^{kl})$. Then find $l_0 \in I^1(x)$ with $\Delta_k(x^{kl_0}) \geq \Delta_k(x^{kl})$ for every $l \in I^1(x)$. If $\Delta_k(x^{kl_0}) \geq 0$ then put $x_{l_0}^k = 0$. After that, assume that x^k is constructed.

It is not difficult to see that all (0, 1)-vectors x^k for $k \in I$ are distinct, and therefore, the neighborhood $N_0(x)$ contains precisely m elements.

Thus, the proposed algorithm for constructing an approximate solution to the problem of maximizing a pseudo-Boolean function $f(x)$ amounts to the standard local search algorithm with the neighborhood $N_0(x)$. The algorithm starts with the solution x^0 obtained by calculating an upper bound and stops having constructed a locally optimal solution x_0.

6. Branch-and-bound algorithm and computational experiments

6.1. The algorithm

Summarizing the above, we can represent the branch-and-bound algorithm for the problem of maximizing the pseudo-Boolean function under study as follows.

The algorithm consists of a finite number of steps. In the first step, we construct a locally optimal solution which is used as the initial record solution.

On the subsequent steps of the algorithm we have a partial solution y, an ordering vector $(i_1, \ldots, i_q)$ with $1 \leq q \leq m$, and record solution x^0. Each step is the inspection of $P(y)$ in order to determine whether there is a better solution than the record.

1. (**First step**)

1.1. Set $q = 0, y = \emptyset$
1.2. Compute the upper bound $H(y)$ and the solution $x(y)$
1.3. Apply to $x(y)$ the local search procedure yielding a local optimal solution x_0, which is the initial record solution x^0.
1.4. If $H(y) \leq f(x^0)$ then goto 3. Otherwise compute the index $i(y)$. Set $q = 1, i_q = i(y), y_{i_q} = 1$. Goto 2.1.

2. (**The main step**)

2.1. If $q = m$ then compute value $f(y)$. If $f(y) > f(x^0)$ then set $x^0 = y$. Goto 2.3.
2.2. If $q < m$ then compute the upper bound $H(y)$, the solution $x(y)$ and value $f(x(y))$. If $f(x(y)) > f(x^0)$ then set $x^0 = x(y)$. If $H(y) > f(x^0)$ then goto 2.4. If $H(y) \leq f(x^0)$ then goto 2.3.
2.3. If $y_{i_k} = 0$ for every k with $1 \leq k \leq q$ then goto 3. Otherwise determine the largest index k with $1 \leq k \leq q$ for which $y_{i_k} = 1$. Set $q = k, y_{i_q} = 0$. Goto 2.1.
2.4. Compute the index $i(y)$. Set $q = q+1$, $i_q = i(y)$, $y_{i_q} = 1$. Goto 2.1.

3. (**Stop**)

Once the algorithm stops, from the resulting (0, 1)-vector x^0 we construct the corresponding optimal noncooperative solution $(X^*, \overline{Z}^*)$ to the competitive facility location problem and calculate the values of the objective functions of the Leader and the Follower on this solution.

6.2. Computational results

Let us present the results of computation experiments with the proposed algorithm. The purpose of the experiment is the evaluation of the computational capabilities of the algorithm and, in particular, the estimation of the number of steps of the algorithm on some examples. In addition, it is important to find out how the initial record received by a standard local search algorithm differs from the optimal solution. We ran calculations on a PC Intel Core i3, 3.1 GHz and to solve the arising integer programming problems used the Microsoft Solver Foundation package.

We solved the problem of competitive location on a network. In this problem the set of possible facility locations I and the set of consumers J coincide with the set of vertices of a weighted network. To each edge of this network we assign a weight called the length of the edge, while the preferences of consumer $j \in J$ are determined by the lengths of the shortest paths from the vertex j to all other vertices. If the lengths of paths coincide then the vertex with a smaller index is preferred. Moreover, for each vertex $i \in I$ we are given three quantities b_i, f_i, and g_i, equal respectively to the revenue received by serving consumer i and the fixed cost for the Leader and the Follower to open facility i.

We solved examples of class R2 taken from the benchmark library [15]. Fig. 1 depicts the "circular" network, corresponding to this class of problems, with two rings, 25 vertices, and with edge lengths specified.

We considered two series of examples with 25 instances in each series. The instances in the first series are of subclass R2d, in

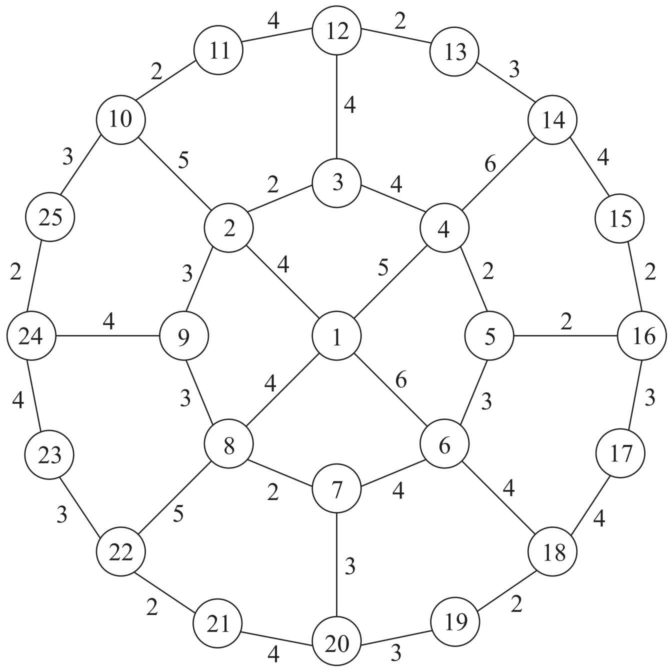

Fig. 1. Circular network with two rings and 25 vertices.

which the edge lengths are fixed, while in the second series, of subclass R2r, in which the edge length is a random variable uniformly distributed on [0.1,1]. In both subclasses the weights b_i, f_i, and g_i are integer-valued random variables uniformly distributed on [6,12], [25, 30], and [15, 20] respectively.

For each example considered of subclasses R2d and R2r we present in Tables 1 and 2 respectively:

- the value L^0 of the objective function of problem (**L**,**F**) on the feasible noncooperative solution $(X^0,\overline{Z}^0)$ obtained by the local search algorithm as the initial record solution;
- the value F^0 of the objective function of problem **F** on the optimal solution $\overline{Z}^0$;
- the indices of the unit components of the vector $x^0=(x_i^0)$;
- the indices of the unit components of the vector $\overline{z}^0=(\overline{z}_i^0)$;
- the value L^* of the objective function of problem (**L**,**F**) on the optimal noncooperative solution $(X^*,\overline{Z}^*)$ obtained by the branch-and-bound algorithm;
- the value F^* of the objective function of problem **F** on the optimal solution $\overline{Z}^*$;
- the indices of the unit components of the vector $x^*=(x_i^*)$;
- the indices of the unit components of the vector $\overline{z}^*=(\overline{z}_i^*)$;
- the number of steps N of the branch-and-bound algorithm;
- the number of steps N_0 of the algorithm until the last change of the record solution;
- the time t taken by the branch-and-bound algorithm, in seconds.

It is clear from these tables that for the instances we considered the number of steps needed for the branch-and-bound algorithm on average is below 100,000, which enables us to find the optimal solution to the problem in about 15 min. Since the complexity of a single step of the algorithm is comparable to the complexity of calculating the value of the objective function of the problem (**L**,**F**) at an feasible noncooperative solution $(X,\tilde{Z})$ for a given solution X then the complexity of the branch-and-bound algorithm for the above examples is essentially less than the complexity of the exhaustive search procedure. In the resulting optimal solutions the Leader opens three facilities on average, while the Follower opens two facilities. At the same time, there are examples with optimal solutions in which the Follower does not open any facilities. This means in particular that an insignificant change of the initial data of this problem can lead to a significant change of the structure of the resulting optimal solution.

Observe also that essentially in all examples the initial record solution obtained by the local search algorithm differs substantially from the optimal solution. Therefore, one direction for improvement to this computational scheme is to refine the algorithms for constructing the initial record solution, whose calculation should not stop at the first locally optimal solution.

Table 1
Optimal solutions for examples of subclass R2d.

Example code	L^0	F^0	x^0	$\overline{z}^0$	L^*	F^*
R2d-01	1	62	(2, 5, 12, 19)	(7, 10, 16)	12	58
R2d-02	24	31	(2, 5, 12, 16, 18, 24)	(7)	24	31
R2d-03	6	75	(5, 12, 20)	(8, 16, 19)	12	57
R2d-04	12	91	(2, 13, 20)	(6, 8, 25)	18	52
R2d-05	25	61	(2, 5, 15, 19)	(7, 11)	25	61
R2d-06	0	205		(2)	14	36
R2d-07	19	37	(2, 5, 8, 13, 16)	(10, 21)	22	34
R2d-08	0	196		(7)	4	47
R2d-09	11	89	(7, 13, 16)	(3,20)	15	56
R2d-10	24	34	(2, 7, 16, 22, 25)	(3,6)	26	24
R2d-11	8	65	(6, 9, 12, 22)	(4,7,11)	22	65
R2d-12	15	63	(2, 12, 16, 21)	(5,7,10)	27	67
R2d-13	3	114	(12,20)	(8,10,17)	10	32
R2d-14	13	33	(2,13,16,19,24)	(7,25)	21	11
R2d-15	6	92	(2,4,22)	(7,15,25)	23	49
R2d-16	10	25	(2,5,10,12,20)	(8,18)	12	26
R2d-17	11	51	(2,13,20,22,25)	(6)	15	64
R2d-18	11	97	(16,19,22)	(4,9)	24	36
R2d-19	10	91	(11,16,20)	(4,8,19)	18	97
R2d-20	15	88	(2,12,20)	(6,8,10)	15	88
R2d-21	1	88	(2,16,19)	(4,11,20)	12	64
R2d-22	2	51	(2,5,7,12)	(10,16,20)	17	11
R2d-23	24	62	(2,16,20)	(9,12,18)	32	50
R2d-24	10	36	(2,5,10,13,22)	(9,18)	22	33
R2d-25	21	51	(2,5,14,22)	(10,16,20)	25	65

Example code	x^*	$\overline{z}^*$	N	N_0	t
R2d-01	(5,9,12,22)	(2,7,16)	84,892	2	839
R2d-02	(2,5,12,16,18,24)	(7)	71,473	1	697
R2d-03	(2,5,12,24)	(16,20)	159,262	159,239	1420
R2d-04	(2,5,13,16,22)	(20,25)	150,968	54,938	1373
R2d-05	(2,5,15,19)	(11,20)	101,793	82,221	853
R2d-06	(2,7,13,16,21)	(5,10)	129,437	30,855	1210
R2d-07	(2,7,13,16,22)	(5,10)	102,347	29,595	960
R2d-08	(2,5,13,20)	(8,18,25)	267,653	133,379	2473
R2d-09	(2,13,16,19)	(7,25)	180,342	10,122	1557
R2d-10	(2,7,16,18,22,25)	(3)	51,484	48,293	396
R2d-11	(2,7,13,22)	(6,25)	284,594	248,070	2742
R2d-12	(2,7,18,21)	(6,11)	127,648	84,537	1056
R2d-13	(3,5,11,15,19,24)	(8)	155,202	33,740	1280
R2d-14	(2,7,13,16,22,25)	(5)	150,847	91,468	1314
R2d-15	(2,5,20,25)	(8,11,16)	118,749	23,820	1052
R2d-16	(2,5,12,20,22)	(10,18)	196,573	63,871	1714
R2d-17	(4,20,22,25)	(2,5)	161,766	70,610	1317
R2d-18	(2,6,13,16,22)	(20,25)	165,496	104,432	1466
R2d-19	(2,7,21)	(6,11)	178,326	167,936	1672
R2d-20	(2,12,20)	(6,8,10)	162,027	1	1587
R2d-21	(5,9,15,22)	(3,20)	321,187	111,143	2905
R2d-22	(2,7,12,16,22,25)	(5)	184,187	92,477	1772
R2d-23	(2,6,16,24)	(12,20)	115,376	104,170	1017
R2d-24	(2,5,10,13,20)	(8,18)	93,815	93,055	740
R2d-25	(2,7,14,22)	(5,10)	106,321	20,935	925

Table 2
Optimal solutions for examples of subclass R2r.

Example code	L^0	F^0	x^0	$\bar{z}^0$	L^*	F^*
R2r-01	5	177	(12)	(2)	13	64
R2r-02	3	110	(9,16,22)	(3,7)	28	74
R2r-03	8	96	(6,12,20)	(4,7,10)	14	48
R2r-04	3	69	(4,10,16,23)	(2,6)	8	133
R2r-05	2	125	(14,19,23)	(9,16)	18	122
R2r-06	14	120	(11,15,24)	(1)	31	22
R2r-07	2	129	(2,24)	(9,12)	3	105
R2r-08	22	51	(8,12,16,20)	(2,17,24)	23	54
R2r-09	0	203		(10)	12	121
R2r-10	11	138	(10,24)	(1)	22	42
R2r-11	13	121	(16,19,24)	(4)	18	74
R2r-12	0	221		(4)	10	89
R2r-13	1	89	(2,12,18,22)	(6,15,25)	28	47
R2r-14	14	94	(8,16,20)	(1,19,22)	19	86
R2r-15	4	79	(2,17,21,22)	(16,25)	7	101
R2r-16	0	214		(8)	21	108
R2r-17	0	121	(15,21)	(5,20)	20	31
R2r-18	30	26	(4,6,12,20,24)	(5,8)	32	35
R2r-19	0	212		(16)	30	9
R2r-20	21	76	(12,16,19,22)	(1)	21	76
R2r-21	11	45	(2,4,8,20)	(13,18,24)	19	60
R2r-22	2	104	(16,21,25)	(4,7)	12	39
R2r-23	21	75	(10,16,22)	(5,8)	23	79
R2r-24	0	200		(4)	14	127
R2r-25	7	84	(2,3,13,22)	(8)	16	66

Example code	x^*	$\bar{z}^*$	N	N_0	t
R2r-01	(10,12,16,21,24)	(6)	159,873	157,746	1031
R2r-02	(8,10,16)	(7,13,23)	38,199	29,415	307
R2r-03	(6,8,12,20,23)	(4,10)	9981	6226	88
R2r-04	(10,23)	(1)	66,415	39,302	449
R2r-05	(10,16,22)	(4)	20,5543	205,326	1095
R2r-06	(1,6,12,15,18,21)	(9)	14,991	2450	122
R2r-07	(4,16,20)	(6,12)	75,802	70,901	651
R2r-08	(8,12,16,21)	(2,18,24)	47,847	12775	367
R2r-09	(2,18)	(4,10)	45,362	42,230	343
R2r-10	(8,15,18,20,24)	(3)	148,229	81,833	1275
R2r-11	(4,18,20,24)	(1,15)	37,006	20,294	285
R2r-12	(4,11,16,18)	(3,22)	99,781	89,652	1002
R2r-13	(6,10,16,22,24)	(13,19)	8589	4587	85
R2r-14	(8,16,19)	(1,6,22)	26,958	13923	255
R2r-15	(2,20,22)	(16,25)	74,184	50,088	522
R2r-16	(14,22,24)	(3)	25,118	24,851	166
R2r-17	(6,9,10,16,21)	(13)	63,183	54,431	455
R2r-18	(4,8,12,20,24)	(5)	9312	1102	59
R2r-19	(4,10,13,17,20,22)	(6)	9754	46	95
R2r-20	(12,16,19,22)	(1)	18,318	1	115
R2r-21	(2,4,22)	(9,13,20)	36,109	23,380	315
R2r-22	(4,10,18,21,24)	(9,14)	112,230	48263	726
R2r-23	(6,10,22)	(8,16)	30,521	29036	195
R2r-24	(14,20)	(1)	76,574	76,475	613
R2r-25	(10,16,20,22)	(2,17)	105,874	104,483	716

7. Conclusions

We study a new competitive facility location problem stated as a discrete bilevel programming problem. We introduce the concept of optimal solution for this problem and propose a branch-and-bound algorithm for seeking the optimal solution.

We believe that the proposed mathematical model is an important, in terms of applications, generalization of the classical facility location problem. This model has a great potential for application and can be used to explore a variety of practical situations where the decision on the placement of objects is taken not by a single party, and herewith the decision and the result of each party depends on the decisions of the others. A possible industry setting for the proposed model described in [16]. We consider a situation of decision making by a company which attempts to win a market share. We assume that the company releases its products to the market under the competitive conditions that another company is making similar products. Both companies can vary the kinds of their products on the market as well as the prices in accordance with consumer preferences. Each company aims to maximize its profit.

In the competitive facility location problem, we assume that the Leader knows the objective function of the Follower, and the Follower makes decisions in accordance with his objective. However, in practical situations, the decision of the Follower can be affected by other factors, and he can make a decision than was unexpected by the Leader. As a result, the decision taken by the Leader is not the best possible but not the worst. The introduction of uncertainty in the problem, in particular, in the behavior of the Follower, is an important way to develop the above model. But so far, we are not aware of any specific proposals in this regard.

The competitive facility location problem is complex to calculate, like other bilevel integer programming problems. The exhaustive search is a universal procedure for solving these problems. Our computational experiments with problems of low dimension (25 variables of the pseudo-Boolean function) show that the above branch-and-bound algorithm has significant advantages over the exhaustive search procedure. As the algorithm runs, the proposed upper bound enables us to discard a large part of subsets considered. As a result, the branch-and-bound algorithm enables us to find optimal solutions to these examples within 15 min.

References

[1] Krarup J, Pruzan PM. The simple plant location problem: survey and synthesis. European Journal of Operational Research 1983;12:36–81.
[2] Beresnev VL. Discrete location problems and polynomials of Boolean variables. Novosibirsk: Sobolev Institute of Mathematics Press; 2005 [in Russian].
[3] Stackelberg H. The theory of the market economy. Oxford: Oxford University Press; 1952.
[4] Dempe S. Foundations of bilevel programming. Dordrecht: Kluwer Academic Publishers; 2002.
[5] Hanjoul P, Peeters D. A facility location problem with client's preference orderings. Regional Science and Urban Economics 1987;17:451–73.
[6] Dobson G, Karmarkar U. Competitive location on network. Operations Research 1987;35:565–74.
[7] Hakimi SL. Locations with spatial interactions: competitive locations and games. In: Michandani PB, Francis RL, editors. Discrete location theory. New York: John Wiley and Sons; 1980. p. 439–78.
[8] Campos Rodríguez CM, Moreno Pérez JA. Multiple voting location problems. European Journal of Operational Research 2008;191:437–53.
[9] Alekseeva E, Kochetova N, Kochetov Y, Plyasunov A. Heuristic and exact methods for the discrete $(r|p)$-central problem. In: EvoCOP 2010, LNCS 6022. Heidelberg: Springer-Verlag; 2010. p. 11–22.
[10] Campos Rodríguez CM, Santos Peñate DR, Moreno Pérez JA. An exact procedure and LP formulations for the leader–follower location problem. Top 2010;18:97–121.
[11] Mitten LG. Branch-and-bound method: general formulation and properties. Operations Research 1970;18:24–34.
[12] Hammer PL, Rudeanu S. Pseudo-Boolean programming. Operations Research 1969;17:233–61.
[13] Beresnev VL. Upper bounds for objective functions of discrete competitive facility location problems. Journal of Applied and Industrial Mathematics 2009;3:419–32.
[14] Beresnev VL, Mel'nikov AA. Approximate algorithms for the competitive facility location problems. Journal of Applied and Industrial Mathematics 2011;5:180–90.
[15] 〈http://www.math.nsc.ru/AP/benchmarks/〉.
[16] Beresnev VL, Suslov VI. Mathematical model of market competition. Journal of Applied and Industrial Mathematics 2010;4:147–57.