Working paper
August 23, 2024

DC programming approach for solving a class of bilevel partial facility interdiction problems

Le Thi Hoai An[1], **Pham Thi Hoai**[2,✉] and **Dinh Tuan Cuong**[2]

Abstract. We propose a new approach based DC programming for finding a solution of the partial facility interdiction problem that belongs to the class of bilevel programming. This model was first considered in the work of Aksen et al. [1] with a heuristic algorithm named multi-start simplex search (MSS). However, because of the big number of the subproblems called during the search, the running time of MSS increases rapidly as the number of facilities grows. To overcome this limitation, our new algorithm based DC programming is able to shorten the computation time significantly, especially for the case of high number of nodes. This efficiency is illustrated by numerical experiments for a lot of test instances up to 20 facilities.

Keywords: Network interdiction problems; Partial facility interdiction; Bilevel programming; Mixed integer bilevel programming.

1. Introduction

1.1. Network interdiction problem

Network interdiction problem was first considered in the early 1960s with the work of Wollmer [4, 5]. This topic has been received a lot of attention up to now because of numerous applications for many disciplines in real life such as security, military, industry, service, etc. One can see [17, 16, 2, 3] and references therein for more details.

The basic and simple interdiction problem involves a defender and an attacker who have opposite targets with a system that usually described by a network. The defender wants to protect the system and to optimize his objective related to the system. In contrast, the other one emphasizes to ruin the structure of this system for his converse aim. In order to do that he chooses suitable arcs (components) of the network to destroy fully or partially. These actions are corresponding to full interdiction and partial interdiction problems, respectively. Depending on the components or the design of the system, the attacker may interdict the whole arc or a part of each arc, for example, a bridge or a pipe line may be full disrupted but for goods or services supply line it can be blocked partially. Moreover, both defender and attacker may be limited by some constraints of resources, e.g., budget constraints.

Actually, the network interdiction problem described above is a special case of a Stackelberg game [8] with two players corresponding to the defender and the attacker of the considered interdiction model. And the mathematical model for this problem is a bilevel

✉ Corresponding author.
[1] LGIPM, Département IA, Université de Lorraine, F-57000, Metz, France & Institut Universitaire de France (IUF), Paris, France. Email address: hoai-an.le-thi@univ-lorraine.fr
[2] Faculty of Mathematics and Informatics, Hanoi University of Science and Technology. Email address: hoai.phamthi@hust.edu.vn, cuong.dinhtuan153@gmail.com

programming min-max or max-min. The more complicate models of tri-level programming min-max-min or max-min-max can be used as the defender and the attacker give actions sequentially three times [16, 18].

Some well-known problems in this field can be mentioned, for example, the shortest path interdiction, maximum flow interdiction, minimum cost flow interdiction, facility interdiction, etc [16]. These kinds of problems can be helpful for warfare or anti-terror as finding important civilian or military infrastructure to attack/protect and therefore can reduce the enemy's fighting power as much as possible. Some papers that concern about this problem such as [6, 7]. The competition in industry or service are able to cast as interdiction models as well, for instances, see [9, 10, 1, 11].

1.2. Problem formulation and motivation

In this paper, we consider one of important applications of network interdiction for services that is the partial facility interdiction problem. Compared to previous studies in this research line [12, 13, 14, 15], partial facility interdiction decisions proposed by Aksen et al [1] are integrated for the first time into a median type network interdiction problem with capacitated facilities and outsourcing option. Let us recall the description of the partial facility interdiction problems given by Aksen et al. [1]. Considering the two agents that are in the position of the attacker and the defender of a system involving capacitated facilities numbered from 1 to m and customers indexed from 1 to n with the following data:

d_{ij}: the distance between customer i and facility j, $i = 1, ..., n; j = 1, ..., m$.

c_d: the shipment fee per unit demand and per unit distance.

c_p: the outsourcing cost per unit demand.

a_i: the demand of customer i, $i = 1, ..., n$.

b_j: the cost for interdicting facility j in full, $j = 1, ..., m$.

b_{tot}: the budget for interdiction.

q_j: the capacity of facility j, $j = 1, ..., m$.

The attacker finds the facilities to destroy within a given budget b_{tot}. The defender then has to control the remaining system with surviving facilities so that all customer's demands are satisfied and minimize the cost simultaneously. The cost is composed by the shipment fee and the outsourcing cost in case of some customer i does not receive enough as requirement a_i. The reason for outsourcing may be that some facilities are restricted by capacity or probably damaged at some portion by the attacker and therefore cannot be able to achieve customer's demands fully. The outsourcing service is provided by the third party with the cost c_p depending on demand not distance.

In the paper of Aksen et al. [1] the author first imposed the rule "single sourcing" to the system, i.e., "a demand node is served either by one and only one facility with sufficient capacity or by the external supplier-whichever is more cost-efficient". This situation is realistic for some circumstance like "customer-centric service systems" [1] since "without this restriction optimal solutions might have a retailer receive many deliveries of the same product (each for, conceivably, a very small amount of the product). Clearly, from a managerial point of view, restricting deliveries to come from only one warehouse is a more

appropriate delivery strategy" [19]. However, according to the defender's point of view, this assumption may waste unused capacity of some facilities as well as increase the final cost. This phenomena is also studied in [1] as the authors consider "multi-sourcing" case of PFIP. The numerical results obtained by a heuristic algorithm multi-start simplex search (MSS) proposed by [1] show that the defender saves about 0.86% to 0.94% the total cost if he applies "multi-sourcing" rule for the system. Moreover, with this assumption the attacker also reduces the consuming time to find the optimal solution about 248 times compared to the "single-sourcing" case, [1]. The reason is in the mathematical models corresponding to each case. In particular, the PFIP with "single-sourcing" rule belongs to the class of mixed integer bilevel linear programming:

$$\max_{S} \; Z(S) = \min_{U} c_d \sum_{i\in\mathbf{I}}\sum_{j\in\mathbf{J}} a_i d_{ij} U_{ij} + c_p \sum_{i\in\mathbf{I}} a_i (1 - \sum_{j\in\mathbf{J}} U_{ij}) \tag{PFIPS}$$

$$\text{s.t.} \; \sum_{i\in\mathbf{I}} a_i U_{ij} \le (1 - S_j) q_j \quad j \in \mathbf{J}, \tag{1.1}$$

$$\sum_{j\in\mathbf{J}} U_{ij} \le 1 \quad i \in \mathbf{I}, \tag{1.2}$$

$$U_{ij} \in \{0,1\} \quad i \in \mathbf{I}, j \in \mathbf{J}. \tag{1.3}$$

$$\text{s.t.} \; \sum_{j\in\mathbf{J}} b_j S_j \le b_{tot}, \tag{1.4}$$

$$0 \le S_j \le 1, j \in \mathbf{J}, \tag{1.5}$$

while the formulation of PFIP "multi-sourcing" is a simple bilevel linear programming

$$\max_{S} \; Z(S) = \min_{U} c_d \sum_{i\in\mathbf{I}}\sum_{j\in\mathbf{J}} a_i d_{ij} U_{ij} + c_p \sum_{i\in\mathbf{I}} a_i (1 - \sum_{j\in\mathbf{J}} U_{ij}) \tag{PFIPM}$$

$$\text{s.t.} \; \sum_{i\in\mathbf{I}} a_i U_{ij} \le (1 - S_j) q_j \quad j \in \mathbf{J}, \tag{1.6}$$

$$\sum_{j\in\mathbf{J}} U_{ij} \le 1 \quad i \in \mathbf{I}, \tag{1.7}$$

$$0 \le U_{ij} \le 1 \quad i \in \mathbf{I}, j \in \mathbf{J}. \tag{1.8}$$

$$\text{s.t.} \; \sum_{j\in\mathbf{J}} b_j S_j \le b_{tot}, \tag{1.9}$$

$$0 \le S_j \le 1, j \in \mathbf{J}, \tag{1.10}$$

In the above problems, variable S_j presents the portion of capacity at facility j interdicted by the attacker while U_{ij} is the percentage of capacity at facility j serving for customer i.

It is clear that (PEIPS) is more difficult than (PFIPM) since the binary variables in the lower level. Therefore, by the above arguments, it is natural for the attacker choosing model (PFIPM) instead of (PFIPS). This choice is not only capable to reduce the solving time a lot compared to (PFIPS) but also consistent to the popular operation "multi-sourcing" between the customers and facilities. For the defender, after knowing the interdiction solution ($\mathbf{S}$) from the attacker, he can solve the lower level problem to find his best solution matching with "multi-sourcing" or "single-sourcing" system.

Hence, in this paper, we first approach PFIP with (PFIPM) model and then consider (PFIPS) if the defender wants to use "single-sourcing" rule for operating.

We know that problem (PFIPM) is a **linear bilevel programming (LBP)** that has been thoroughly studied in both theoretical and algorithmic aspects in literature. So, firstly, **let us review some main points for a general LBP**. One can also see [27, 28, 26, 25] and the references therein for more details. In qualitative research, the NP-hardness of (LBP) can be derived from the results in the paper [21] (1985) for the first time. Not only that, in [22], the authors prove that (LBP) is strongly NP-hard. Even the checking the locality of a given solution of (LBP) is also an NP-hard problem, see [23]. Other important properties related to (LBP) are nonconvex, nonsmooth that challenge us to find its global optimal solutions. Regarding the algorithmic point of view, one of the first algorithms for solving (LBP) is the K-best algorithm proposed by [30]. The idea is similar to simplex method for linear programming, the algorithm searches over the vertices of the feasible set to find a solution. Nevertheless, at each step of algorithm, this method requires computing all the adjacent extreme points and consequently it costs expensive effort. An other popular approach for solving (LBP) is collapsing the two levels to get a "single-level" optimization problem by using KKT conditions[**?**] for the lower level. The obtained reformulation is a mathematical problem with complementary contraints (MPCC) that is nonlinear and still difficult. To deal with the nonlinearity, the big-M technique is suggested to transfer MPCC to be a mixed integer linear programming. However, finding a corrected and large value big-M leads to other difficult problems as well because "it is not a free lunch" [31]. To avoid this obstacle the branch and bound scheme can be applied to MPCC, see [29]. But this method is usually takes time a lot as the size of considered problem increases. In such a situation, one should consider some heuristic methods that may not be confirmed to achieve to global optimization but give a good enough solution in short computation time. In [32], by using strong duality property, (LBP) is rewrited as a nonconvex single-level optimization problem with nonlinear and nonconvex constraints. The authors then use penalty technique to reformulate it to be a smooth optimization problem with nonconvex quadratic objective and linear constraints. Finally, they apply penalty alternating direction method to find a solution of (LBP). This approach, however, copes with or depends on the choice of penalty coefficient during the search.

1.3. Contribution

To overcome the drawbacks of mentioned methods in literature, we propose new algorithms based DC programming for solving (PFIPM) and (PFIPS) in this work. In fact, DC programming plays an important role in theory of optimization. It concerns about nonconvex optimization problems that have a lot of applications in real life [33, 34]. One of efficient approaches for solving a DC programming is DCA by P.D. Tao [40]. This method has been then extensively developed by Tao and An [37]. DCA can work with nonconvex and nonsmooth optimization problems. The advantage of DCA has been showed by many studies for applicative problems, see [33] and references therein.

In summarize, our new algorithms based DC programming can solve (PFIP) model for both multi-sourcing and single-sourcing cases and have the following points:

- Our algorithms can be easy implemented without guessing or computing any supported coefficients like big-M or penalty coefficient.

- Our algorithm can work with large size (PFIP) in very short time. This efficiency is illustrated by rich computational results for 170 instances compared to MSS algorithm [1] with up to 20 facilities.

- The obtained numerical results show that our method provides better solution in shorter computation time for both (PFIPM) and (PFIPS) compared to MMS algorithm [1].

The rest of the paper is organized as follows. In section 2 we review the DCA scheme for solving a DC programming. Our new algorithms based DCA for finding a solution of (PFIPM) and (PFIPS) are proposed in section 3. After that we report and analyse the numerical results in section 4. Finally, the paper is closed with some conclusions in section 5.

2. DC programming and DCA

In this section, we recall some basic concepts used for DC programming and DCA scheme, see e.g. [37, 36, 35, 33].

If the function u is defined in $C \subset \mathbb{R}^n$ then we can extend it to be a function that defined in $\mathbb{R}^n$ by setting $u(x) = +\infty$ for $x \notin C$. Denoting

$$\text{dom}u = \{x \in \mathbb{R}^n \mid u(x) < +\infty\}.$$

$u : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is call proper function if $\text{dom}u \neq \emptyset$ and $u(x) > -\infty$ for all $x \in \mathbb{R}^n$.
$\Gamma_0(\mathbb{R}^n)$ denotes for the set of all lower semicontinuous, convex and proper functions over $\mathbb{R}^n$.
The indicator function of a nonempty convex set C is defined by

$$\chi_C(x) = \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{otherwise} \end{cases}.$$

The function

$$\theta(x) = \max\{\left\langle a^i, x \right\rangle - \alpha_i, i = 1, \ldots, m\} + \chi_C(x),$$

where $a^i \in \mathbb{R}^n, \alpha_i \in \mathbb{R}$, $i = 1, \ldots, m$; C is a convex set of $\mathbb{R}^n$, is a convex function and called a polyhedral function.
We define the conjugate function u^* of u by

$$u^*(y) = \sup\{\langle x, y\rangle - u(x) \mid x \in \mathbb{R}^n\}, \text{ for } y \in \mathbb{R}^n.$$

$p \in \mathbb{R}^n$ is subgradient of a proper function u at x^0 ($x^0 \in \text{dom}f$) if

$$\left\langle p, x - x^0 \right\rangle \leq u(x) - u(x^0) \text{ for all } x \in \mathbb{R}^n.$$

The set of all subgradient of u at x^0 is called subdifferential of u at x^0 and denoted by $\partial f(x^0)$.
x^* is called critical point of $g - h$ if $\partial g(x^*) \cap \partial h(x^*) \neq \emptyset$.
The standard DC programming we consider can be stated as follows

$$\alpha = \inf\{f(x) = g(x) - h(x) \mid x \in \mathbb{R}^n\}, \tag{P}$$

where $g, h : \mathbb{R}^n \to \mathbb{R} \cup \{-\infty, +\infty\} \in \Gamma_0(\mathbb{R}^n)$. In the sequel, we admit the convention $+\infty - (+\infty) = +\infty$ and assuming that

$$\text{dom}g \subset \text{dom}h \ \text{dom}h^* \subset \text{dom}g^*$$

without loss of generality. This assumption helps to avoid the trivial case that $\alpha = -\infty$.

According to [36], any general DC programming can be transformed to be a standard DC programming like (P) by using the indicator function or penalty technique. And problem (P) can be rewrote as

$$\alpha = \inf\{h^*(y) - g^*(y) \mid y \in \mathbb{R}^n\}. \tag{D}$$

The main idea of DCA is constructing the two sequences $\{x^k\}$ $\{y^k\}$ satisfies

(i) The sequences $\{(g-h)(x^k)\}$ and $\{(h^*-g^*)(y^k)\}$ are decreasing.

(ii) Each limit point x^* (y^*, resp.) of $\{x^k\}$ ($\{y^k\}$, resp.) is a critical point of $g-h$ (h^*-g^*, resp.).

Below is DCA scheme for solving problem (P) (see [36, 37]).

DCA scheme for (P)

Initialization: choosing an initial point $x^0 \in \text{dom}g$, $k \leftarrow 0$.

Step 1: computing $y^k \in \partial h(x^k)$

Step 2: finding $x^{k+1} \in \partial g^*(y^k) = \text{argmin}\{g(x) - \langle x, y^k\rangle \mid x \in \mathbb{R}^n\}$

Step 3: $k \leftarrow k+1$ and return Step 1 until convergence of $\{x^k\}$

The convergent properties of DCA can be summarized in theorem below.

Theorem 2.1. *(i) In DCA, the sequences $\{(g-h)(x^k)\}$ and $\{(h^*-g^*)(y^k)\}$ are decreasing.*

(ii) If α is finite then every limit point x^* (y^*, resp.) of $\{x^k\}$ ($\{y^k\}$, resp.) is a critical point of $g-h$ (h^*-g^*, resp.).

(iii) DCA is finite convergent if g or h is a polyhedral function.

Proof. One can find the details of the proof in [37]. □

3. New algorithms based DCA for solving (PFIPM) and (PFIPS)

Firstly, we consider problem (PFIPM) and reformulate it to be a standard form of DC programming.

Let

$$C = \{S \in \mathbb{R}^m \mid \sum_{j\in\mathbf{J}} b_j S_j \le b_{tot}, 0 \le S_j \le 1, j \in \mathbf{J}\}.$$

Note that, for a fixed $S \in C$, $Z(S)$ is the optimal value of the lower level:

$$Z(S) = \min_U c_d \sum_{i\in\mathbf{I}}\sum_{j\in\mathbf{J}} a_i d_{ij} U_{ij} + c_p \sum_{i\in\mathbf{I}} a_i (1 - \sum_{j\in\mathbf{J}} U_{ij}) \tag{LVM}$$

$$\text{s.t.} \quad \sum_{i\in\mathbf{I}} a_i U_{ij} \le (1-S_j) q_j \quad j \in \mathbf{J}, \tag{3.1}$$

$$\sum_{j\in\mathbf{J}} U_{ij} \le 1 \quad i \in \mathbf{I}, \tag{3.2}$$

$$U_{ij} \ge 0 \quad i \in \mathbf{I}, j \in \mathbf{J}. \tag{3.3}$$

It is easy to see that, for any $S \in C$, LVM is a linear programming with respect to (U_{ij}) and its feasible set is non-empty as well as bounded. Consequently, LVM always admits an optimal solution and so do its duality

$$Z(S) = \max_{x,y} \sum_{j\in\mathbf{J}} x_j(1-S_j)q_j + \sum_{i\in\mathbf{I}} y_i \tag{LVMD}$$

$$\text{s.t. } y_i + x_j a_i \le a_i(c_d d_{ij} - c_p) \quad i \in \mathbf{I}, j \in \mathbf{J}, \tag{3.4}$$

$$x_j, y_i \le 0 \quad i \in \mathbf{I}, j \in \mathbf{J}. \tag{3.5}$$

In (LVMD), x_j $(j \in \mathbf{J})$ and y_i $(i \in \mathbf{I})$ are dual variables for constraints (3.1) and (3.2), respectively.
Setting

$$X = (x_1, ..., x_m, y_1, ..., y_n)^T \in \mathbb{R}^{m+n},$$

and

$$r(S) = ((1-S_1)q_1, ..., (1-S_m)q_m, 1, ..., 1)^T \in \mathbb{R}^{m+n}.$$

Denote the feasible set of problem (LVMD) by $\mathcal{M}$ then $\mathcal{M}$ is independent of S obviously. We transfer (LVMD) to be in the compact form as follows

$$Z(S) = \max_{X\in\mathcal{M}} r(S)^T X. \tag{3.6}$$

Problem (PFIPM) now becomes

$$\max_{S\in C} Z(S) \Leftrightarrow \min_{S\in C} -Z(S). \tag{3.7}$$

Setting

$$H(S) = \begin{cases} Z(S), & \text{if } S \in C, \\ +\infty, & \text{if } S \in \mathbb{R}^m \setminus C. \end{cases}$$

We have the following proposition.

Proposition 3.1. *$H(S)$ is a polyhedral convex function on $\mathbb{R}^m$.*

Proof. Firstly, it is worth noting that for all $S \in C$ the optimal value of problem (3.6) is finite and achieve at some vertex of $\mathcal{M}$ or in other words,

$$H(S) = Z(S) = \max_{X\in V(\mathcal{M})} r(S)^T X = \max_{v\in V(\mathcal{M})} r(S)^T v,$$

where $V(\mathcal{M}) = \{v^1, ..., v^h\}$ is the set of all vertices of $\mathcal{M}$. By the linearity of $r(S)$ with respect to S, we get the desired conclusion lastly.

□

Back to problem (PFIPM), without difficulty we can rewrite (3.7) in the equivalent form of a standard DC programming

$$\min_{S\in\mathbb{R}^m} \chi_C(S) - H(S). \tag{PFIPM-DC}$$

From Proposition 3.1, problem (PFIPM-DC) is a polyhedral DC programming that has finite convergent DCA. Below is DCA scheme for (PFIPM-DC)

Algorithm 1: DCA for (PFIPM-DC)

- **Input**: taking an initial point $S^0 \in C, k \leftarrow 0$ and a tolerance $\epsilon \geq 0$.
- **Output**: a solution of (PFIPM).

Iteration k

Step 1: Computing $T^k \in \partial H(S^k)$ by

$$T_l^k = \begin{cases} -q_l \eta_l^k & \text{for } 1 \leq l \leq m \\ 0 & \text{if } m < l \leq m+n \end{cases},$$

where

$$\eta^k = \arg\max_{\eta \in M} r(S^k)^T \eta.$$

Step 2: $S^{k+1} \in \arg\min_{S \in C} (-\langle S, T^k \rangle)$.

Step 3: **If** $\frac{\|S^{k+1} - S^k\|}{\max(\|S^{k+1}\|, 1)} \leq \epsilon$ **then** STOP and S^{k+1} is a desired solution.
else $k \leftarrow k+1$, **return** to Step 1.

As a result of Theorem 2.1 we obtain the following convergent properties convergence of Algorithm 1 as follows.

Theorem 3.1. *Algorithm 1 is finite convergent for all $\epsilon \geq 0$ and $\{-H(S^k)\}$ is decreasing.*

Remark 3.1. (i) Actually, we are able to embed the formula of the function $Z(S)$ from (LVMD) into (PFIPM) to get a single-level optimization directly as follows

$$Z(S) = \max_{S,x,y} \sum_{j \in \mathbf{J}} x_j (1 - S_j) q_j + \sum_{i \in \mathbf{I}} y_i \tag{PFIPQ}$$

$$\text{s.t. } y_i + x_j a_i \leq a_i (c_d d_{ij} - c_p) \quad i \in \mathbf{I}, j \in \mathbf{J}, \tag{3.8}$$

$$x_j, y_i \leq 0 \quad i \in \mathbf{I}, j \in \mathbf{J}. \tag{3.9}$$

$$\sum_{j \in \mathbf{J}} b_j S_j \leq b_{tot}, \tag{3.10}$$

$$0 \leq S_j \leq 1, j \in \mathbf{J}. \tag{3.11}$$

We see that (PFIPQ) is a nonlinear and nonconvex optimization problem with indefinite quadratic objective function. So finding a solution of (PFIPQ) is not easy. Moreover, the number of variables of (PFIPQ) is $n + 2m$ that is much more than m-the dimension of problem (PFIPM-DC). This deviation is bigger and bigger if number of customers and facilities grow. Hence, we choose (PFIPM-DC) to continue instead of (PFIPQ).

(ii) In Algorithm 1, one can choose $S^0 \in \mathbb{R}^m$ arbitrary instead of adding the condition $S^0 \in C$. Indeed, at iteration 0 we take $T^0 = 0$. From $k = 1$, S^k is a solution of problem $\min_{S \in C} \langle S, -T^k \rangle$ and as a result $S^k \in C$ for all $k \geq 1$.

From practical point of view as argued in section 1, the attacker should find his solution by solving (PFIPM) for both "single-sourcing" and "multi-sourcing" cases. So, after knowing the attacker's interdiction S^* obtained from Algorithm 1, if the defender want to search "single-sourcing" operation for his system, he can just solve the lower level problem of (PFIPS).
Finally, a solution of (PFIPS) can be found by Algorithm 2 below.

Algorithm 2: to find a solution for (PFIPS).

- **Input:** S^* obtained by Algorithm 1.
- **Output:** a solution of (PFIPS)

  Step 1: solving the mixed integer linear programming

$$\min_{U} c_d \sum_{i\in\mathbf{I}}\sum_{j\in\mathbf{J}} a_i d_{ij} U_{ij} + c_p \sum_{i\in\mathbf{I}} a_i (1 - \sum_{j\in\mathbf{J}} U_{ij}) \tag{PFIPS}$$

$$\text{s.t.} \sum_{i\in\mathbf{I}} a_i U_{ij} \le (1 - S*_j) q_j \quad j \in \mathbf{J}, \tag{3.12}$$

$$\sum_{j\in\mathbf{J}} U_{ij} \le 1 \quad i \in \mathbf{I}, \tag{3.13}$$

$$U_{ij} \in \{0,1\} \quad i \in \mathbf{I}, j \in \mathbf{J} \tag{3.14}$$

  to get (U^*_{ij}).

  Step 2: (S^*, U^*) is a desired solution of (PFIPS).

4. Computational results

We generate randomly tested instances for $m \in \{4, 5, 6, ..., 20\}$ and $n = 10m$ by using the same rules in [1]. In particular,

- the location of n nodes are in a sphere of origin $(0, 0)$ and radius 500;
- the location of facilities are settled on equidistant horizontal and vertical lines slicing a square with side 1000 that is concentric with the sphere above;
- All coordinates are rounded to the nearest integer;
- $c_d = 0.1$, $c_p = 100$;
- a_i is randomly chosen from the set $\{5, 10, ..., 100\}$;
- b_j is randomly picked from $\{15000, 16000, ..., 30000\}$;
- calculating $\omega = \sum_j b_j / \sum_i a_i$ và $q'_j = b_j/\omega$ then q_j is obtained by taking q'_j as the nearest integer multiples of 20 such that $\sum_j q_j \ge \sum_i a_i$;
- b_{tot} equals to 30% and 60% of the total full capacities interdiction costs that are corresponding to low and high budget cases.

We generated randomly each size of (m, n) : 5 instances for low interdiction budget and 5 instances for the case of high budget. So the total number of our tested instances is 170. We implemented Algorithm 1, Algorithm 2 and MSS algorithm for solving (PFIPM) and (PFIPS) by coding in Matlab R2014a equipped with Gurobi [41] version 7.5.2.Our PC has the configuration of chip Intel Core i5-3230M 2.60GHz (4 CPUs), 8GB DDR3 based Windows 10 64-bit.

As suggested in [1], the max iterations for three phases 1, 2, 3 of MSS are 100, 100 and 150, respectively . The tolerance $\epsilon = 10^{-7}$ for phase 1 and 2, and $\epsilon = 10^{-8}$ for phase 3. For our algorithms, we limited 100 for max iteration and the tolerance $\epsilon = 10^{-8}$. The solution time for all algorithms to solve one instance is limited in 7200s. The initial point used for MSS and our method are the same, i.e., obtained by procedure $Generat\_V_1$ in [1].

In Table 4.1, 4.2, 4.3 and 4.4 we reported the comparison between Algorithm 2, Algorithm 1 and MSS for solving problem (PFIPS) and (PFIPM) for the two cases of low and high interdiction budgets, respectively. $Z^{(1)}$, $Z^{(2)}$ and $Z^{(MSS)}$ are optimal values obtained by Algorithm 1, Algorithm 2 and MSS algorithm. Gap gives the relative deviation between $Z^{(1)}$ and $Z^{(MSS)}$ or $Z^{(2)}$ and $Z^{(MSS)}$ by the formula

$$\text{Gap} = \frac{Z^{(1)} - Z^{(MSS)}}{Z^{(MSS)}} \text{ for Table 4.1, 4.2}$$

and

$$\text{Gap} = \frac{Z^{(2)} - Z^{(MSS)}}{Z^{(MSS)}} \text{ for Table 4.3, 4.4.}$$

The running time for Algorithm 1, 2 and MSS are denoted by $time^{(1)}$, $time^{(2)}$ and $time^{(MSS)}$ in all tables. Finally, in all tables, the notation "-" means that the instance cannot be solved within limited running time 7200s.

The numerical results for each size (m, n) are computed by taking the average on 5 randomly instances.

m	$Z^{(2)}$	$Z^{(MSS)}$	Gap(%)	$time^{(2)}$ (s)	$time^{(MSS)}$ (s)
4	1.1700e+05	1.1876e+05	-1.48	2.31	107.21
5	1.4576e+05	1.4727e+05	-1.03	3.59	260.18
6	1.6075e+05	1.6277e+05	-1.24	4.38	432.04
7	1.7995e+05	1.8182e+05	-1.02	6.82	690.97
8	2.1447e+05	2.1420e+05	0.12	8.48	1515.92
9	2.1545e+05	2.1664e+05	-0.55	12.35	1872.61
10	2.5916e+05	2.5916e+05	0.00	16.33	4172.59
11	2.5583e+05	2.5458e+05	0.49	24.09	2856.28
12	2.7997e+05	2.7462e+05	1.95	23.75	4267.73
13	2.9700e+05	2.9092e+05	2.09	27.37	-
14	3.3682e+05	3.2818e+05	2.63	38.22	-
15	3.3357e+05	3.2229e+05	3.50	47.34	-
16	3.5534e+05	3.4528e+05	2.91	49.20	-
17	3.9052e+05	3.6204e+05	7.87	77.04	-
18	4.0507e+05	3.8342e+05	5.65	82.08	-
19	4.0778e+05	3.8807e+05	5.08	89.74	-
20	4.3802e+05	4.0979e+05	6.89	126.99	-
AVG	**2.8191e+05**	**2.7411e+05**	**2.85**	**37.65**	**-**

Table 4.1: Low budget for (PFIPS)

m	$Z^{(2)}$	$Z^{(MSS)}$	Gap(%)	$time^{(2)}$ (s)	$time^{(MSS)}$ (s)
4	1.6616e+05	1.6650e+05	-0.21	3.49	157.13
5	1.8454e+05	1.8901e+05	-2.36	5.99	255.50
6	2.3855e+05	2.4078e+05	-0.93	8.54	372.73
7	2.6712e+05	2.6858e+05	-0.54	10.47	704.90
8	3.0559e+05	3.0855e+05	-0.96	11.80	1190.52
9	3.3364e+05	3.3346e+05	0.06	18.47	1493.36
10	3.6092e+05	3.6029e+05	0.18	25.37	2332.46
11	3.9906e+05	4.0013e+05	-0.27	31.39	3859.71
12	4.5757e+05	4.5555e+05	0.44	36.63	-
13	4.6595e+05	4.5809e+05	1.72	32.20	-
14	5.0391e+05	4.9932e+05	0.92	48.08	-
15	5.3740e+05	5.2662e+05	2.05	72.15	-
16	5.8452e+05	5.7426e+05	1.79	86.00	-
17	5.9729e+05	5.8489e+05	2.12	102.55	-
18	6.2670e+05	6.1441e+05	2.00	137.54	-
19	6.8972e+05	6.6765e+05	3.31	136.04	-
20	7.0769e+05	6.8316e+05	3.59	148.16	-
AVG	**4.3684e+05**	**4.3125e+05**	**1.30**	**53.82**	**-**

Table 4.2: High budget for (PFIPS)

m	$Z^{(1)}$	$Z^{(MSS)}$	Gap(%)	$time^{(1)}$ (s)	$time^{(MSS)}$ (s)
4	1.1570e+05	1.1704e+05	-1.15	3.37	39.23
5	1.4712e+05	1.4749e+05	-0.25	5.15	80.83
6	1.6212e+05	1.6188e+05	0.15	7.04	119.91
7	1.7998e+05	1.8058e+05	-0.33	10.62	173.72
8	2.1092e+05	2.1307e+05	-1.01	16.46	239.01
9	2.1668e+05	2.1788e+05	-0.55	20.45	305.82
10	2.5886e+05	2.5822e+05	0.25	31.46	389.43
11	2.5747e+05	2.5555e+05	0.75	40.65	451.43
12	2.7506e+05	2.7730e+05	-0.81	37.36	547.60
13	2.9353e+05	2.9455e+05	-0.35	49.66	840.38
14	3.3585e+05	3.3340e+05	0.74	71.98	971.68
15	3.3080e+05	3.2818e+05	0.80	59.41	1014.04
16	3.5593e+05	3.5330e+05	0.74	70.42	1153.75
17	3.8548e+05	3.8235e+05	0.82	82.73	1254.45
18	4.0342e+05	3.9912e+05	1.08	98.04	1403.90
19	4.0105e+05	3.9827e+05	0.70	87.33	1560.35
20	4.3877e+05	4.2633e+05	2.92	92.82	1750.89

AVG	**2.8051e+05**	**2.7909e+05**	**0.51**	**46.17**	**723.32**

Table 4.3: Low budget for (PFIPM)

m	$Z^{(1)}$	$Z^{(MSS)}$	Gap(%)	$time^{(1)}$ (s)	$time^{(MSS)}$ (s)
4	1.6434e+05	1.6598e+05	-0.99	3.00	55.21
5	1.8422e+05	1.8551e+05	-0.69	5.24	79.54
6	2.3788e+05	2.3739e+05	0.20	7.00	149.72
7	2.6813e+05	2.6816e+05	-0.01	8.29	214.26
8	3.0774e+05	3.0681e+05	0.30	11.58	252.14
9	3.3307e+05	3.3386e+05	-0.24	16.77	335.16
10	3.6137e+05	3.5964e+05	0.48	25.96	468.76
11	3.9798e+05	3.9794e+05	0.01	25.81	602.45
12	4.5665e+05	4.5524e+05	0.31	33.63	818.90
13	4.6396e+05	4.5988e+05	0.89	40.53	1109.14
14	5.0086e+05	5.0091e+05	-0.01	45.86	1353.79
15	5.3762e+05	5.3141e+05	1.17	64.86	1282.98
16	5.8076e+05	5.7587e+05	0.85	78.08	1660.67
17	6.0014e+05	5.9108e+05	1.53	117.06	2079.76
18	6.2891e+05	6.2165e+05	1.17	119.52	1998.27
19	6.8722e+05	6.8079e+05	0.94	144.04	1605.18
20	7.0600e+05	6.8980e+05	2.35	229.25	1996.96
AVG	**4.3629e+05**	**4.3305e+05**	**0.75**	**57.44**	**944.89**

Table 4.4: High budget for (PFIPM)

From the reported information we see that:

- The running time for solving low budget instances is shorter than high budget case for both of methods. This is not difficult to explain since low budget case makes the feasible set of (PIFP) is smaller than high budget case and hence it takes less time.

- MSS is more expensive than our method for both "single-sourcing" and "multi-sourcing" cases. Especially for (PFIPS), the difference in running time between Algorithm 2 and MSS is very huge. With more than 12 facilities MSS cannot stop in limited time 7200s since it has to solve many binary linear subproblems during the search. Although the difference in solution time between Algorithm 1 and MSS for (PFIPM) is less than (PFIPS) but it still large. Specifically, our method is about 16 times faster than MSS.

- In all tables the optimal values of (PFIPS) and (PFIPM) obtained by our method are better than MSS. For single-sourcing rule, the average gap for 85 instances of low budget interdiction is 2.85% and of high budget is 1.3%. For multi-sourcing case these results are 0.5% and 0.75% corresponding.

- In general, our method can solve (PIFP) efficiently in very short time even for large dimension of considered problem. The solution time is around 200s for the large sizes up to 20 facilities and 200 customers. One of the reasons is that the simplicity of DCA applied for (PIFP).

5. Conclusion

In this paper, we consider the partial interdiction facility problem (PIFP) proposed by Aksen et al. [1]. By using an approach based DC programming we propose new algorithms based DCA for solving this model for both cases of single-sourcing and multi-sourcing. The computational results show that our method outperforms MSS algorithm [1] significantly in running time as well as in the quality of obtained solution. The proposed method in this paper should be developed for more general bilevel model in our future research.

References

[1] D. Aksen, S. S. Akca and N. Aras, A bilevel partial interdiction problem with capacitated facilities and demand outsourcing, Computers and Operations Research (2014), 41: 346-358.

[2] R.K. Wood, Deterministic network interdiction. Math. Comput. Model;17:1–18 (1993).

[3] R.K. Wood, Bilevel Network Interdiction Models: Formulations and Solutions 15 February 2011. Wiley Online Library, doi.org/10.1002/9780470400531.eorms0932.

[4] R.D. Wollmer , Some methods for determining the most vital links in a railway network, The Rand corporation RM-3321-ISA (1963).

[5] R.D. Wollmer, Removing arcs from a network, Oper. Res. (1964); 12: 934–940.

[6] D. MacIsaac, Strategic bombing in world war two (1976), New York: Garland Publishing, Inc.

[7] C. Lim, J.C. Smith, Algorithms for discrete and continuous multicommodity flow network interdiction problems (2007), IIE Trans. 39 (1): 15-26.

[8] H. von Stackelberg, The Theory of the Market Economy (William Hodge, London, 1952).

[9] S.D. Cartwright, Supply chain interdiction and corporate warfare, J. Bus. Strat. (2000) 21(2): 30–35.

[10] J.C. Smith, C. Lim, A. Alptekinoglu, New product introduction against a predator: A bilevel mixed-integer programming approach (2009), Nav. Res. Logist. 56: 714–729.

[11] A. Forghani, F. Dehghanian, M. Salari, Y. Ghiami, A bi-level model and solution methods for partial interdiction problem on capacitated hierarchical facilities, Computers & Operations Research (2020) 114, https://doi.org/10.1016/j.cor.2019.104831

[12] R.L. Church, M.P. Scaparra, R.S. Middleton, Identifying critical infrastructure: the median and covering facility interdiction problems. Annals of the Association of American Geographers (2004),94(3):491–502.

[13] F Liberatore, M.P. Scaparra, M.S. Daskin, Hedging against disruptions with ripple effects in location analysis, Omega 2012, 40(1):21–30.

[14] M.P. Scaparra, R.L. Church, Protecting supply systems to mitigate potential disaster: a model to fortify capacitated facilities, Kent Business School working paper no. 209. UK: University of Kent; 2010.

[15] L.V. Snyder, M.P. Scaparra, M.S. Daskin, R.L Church, Planning for disruptions in supply chain networks. Tutorials in Operations Research 2006 [INFORMS], https://doi.org/10.1287/educ.1063.0025

[16] J.C. Smith, M. Prince, J. Geunes, Modern Network Interdiction Problems and Algorithms, In: Pardalos P., Du DZ., Graham R. (eds) Handbook of Combinatorial Optimization (2013) Springer, New York: 1949-1987.

[17] J. C. Smith, Y. Song, A Survey of Network Interdiction Models and Algorithms, European Journal of Operational Research (2019), doi: https://doi.org/10.1016/j.ejor.2019.06.024

[18] S. Sadeghi, A. Seifi and E. Azizi, Trilevel shortest path network interdiction with partial fortification, Computers & Industrial Engineering (2017), 106: 400-411.

[19] J. Bramel, D. Simchi-Levi, The logic of logistics: theory, algorithms, and applications for logistics management.Springer series in operations research (1997), New York: Springer-Verlag.

[20] W. F. Bialas and M. H. Karwan "Two-level linear programming." In: Management Science (1984), 30(8): 1004–1020. doi: 10.1287/mnsc.30.8. 1004.

[21] R. G. Jeroslow, The polynomial hierarchy and a simple model for competitive analysis, In: Mathematical Programming (1985) 32(2): 146–164. doi: 10.1007/BF01586088.

[22] P. Hansen, B. Jaumard and G. Savard, New branch-and-bound rules for linear bilevel programming, In: SIAM Journal on scientific and Statistical Computing (1992), 13(5): 1194–1217, doi: 10.1137/0913069.

[23] L. Vicente, G. Savard and J. Júdice, Descent approaches for quadratic bilevel programming, Journal of Optimization Theory and Applications (1994), 81(2): 379–399, doi: 10.1007/BF02191670.

[24] T. Kleinert, Algorithms for Mixed-Integer Bilevel Problems with Convex Followers, PhD thesis. url: https://opus4.kobv.de/opus4-trr154/frontdoor/index/index/docId/383 (2021).

[25] B. Colson, P. Marcotte, and G. Savard, Bilevel programming: A survey, 4OR (2005) 3(2): 87–107. doi: 10.1007/s10288-005-0071-0.

[26] B. Colson, P. Marcotte, and G. Savard, An overview of bilevel optimization, Annals of Operations Research (2007), 153(1): 235–256, doi: 10.1007/s10479-007-0176-2.

[27] T. Kleinert, J. Manns, M. Schmidt, and D. Weninger, Presolving Linear Bilevel Optimization Problems, Tech. rep. (2021), url: www . optimization-online.org/DB_HTML/2021/03/8286.html.

[28] S. Dempe, . "Bilevel Optimization: Theory, Algorithms, Applications and a Bibliography, Bilevel Optimization: Advances and Next Challenges. Ed. by S. Dempe and A. Zemkoho. Springer International Publishing (2020): 581–672. doi: 10.1007/978-3-030-52119-6_20.

[29] M. Schmidt, A Gentle and Incomplete Introduction to Bilevel Optimization, http://www.optimization-online.org/DB_FILE/2021/06/8450.pdf (2021).

[30] W. F. Bialas and M. H. Karwan, Two-level linear programming In: Management Science (1984), 30(8): 1004–1020, doi: 10.1287/mnsc.30.8. 1004.

[31] T. Kleinert, M. Labbé, F. Plein, and M. Schmidt, There's No Free Lunch: On the Hardness of Choosing a Correct Big-M in Bilevel Optimization, Operations Research (2020), 68(6): 1716–1721. doi: 10. 1287/opre.2019.1944.

[32] T. Kleinert and M. Schmidt, Computing Feasible Points of Bilevel Problems with a Penalty Alternating Direction Method, INFORMS Journal on Computing (2019a), doi: 10.1287/ijoc.2019.0945. Forthcoming.

[33] L.T.H. An, P.D. Tao, DC programming and DCA: thirty years of developments, Math. Program. (2018), Ser. B, 169 (1): 5-68.

[34] H. Tuy, Convex Analysis and Global Optimization, the second edition, Springer International Publishing AG Switzerland (2016).

[35] L.T.H. An, P.D. Tao, The DC (Difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems, Ann. Oper. Res. (2005), 133: 23- 46.

[36] L.T.H. An, P.D. Tao, DC programming in communication systems: challenging problems and methods, Vietnam J. Comput. Sci. (2014), 1: 15-28.

[37] P.D. Tao, L.T.H. An, Convex analysis approach to DC programming: theory, algorithms and applications, Acta Math. Vietnam (1997), Vol. 22(1): 289-355, Dedicated to Hoang Tuy on the occasion of his seventieth birthday.

[38] L.T.H. An, P.D. Tao, DC Programming: Theory, Algorithms and Applications. The State of the Art, 26 pages. Proceedings of The First International Workshop on Global Constrained Optimization and Constraint Satisfaction (Cocos' 02)", Valbonne-Sophia Antipolis, France (2002).

[39] N.C. Nam, P.T. Hoai, A continuous DC programming approach for resource allocation in OFDMA/TDD wireless networks, Computers & Operations Research (2017), 82: 95-101.

[40] P. D. Tao, Algorithms for solving a class of non convex optimization problems. Methods of subgradients, Fermat days 85. Mathematics for Optimization, J. B. Hiriart Urruty (ed.), Elsevier Science Publishers B. V. North-Holland (1986).

[41] *http* : *//www.gurobi.com/*.