Journal of Mathematical Extension
Vol. 16, No. 2, (2022) (10)1-38
URL: https://doi.org/10.30495/JME.2022.1732
ISSN: 1735-8299
Original Research Paper

A Bi-Level Programming Model for Protecting an Important Node in a Network

H. R. Maleki [*]
Shiraz University of Technology

Z. Maleki
Shiraz University of Technology

R. Akbari
Shiraz University of Technology

Abstract. Protecting important nodes in a network against natural disasters, security threats, attacks, and so on is one of the main goals of network planners. In this paper, a new model is presented for protecting an important node (NMPN) in a typical network based on a defensive location problem where the threatening agent (t-agent) can reinforce its power at some nodes. The NMPN is a bi-level programming problem. At the upper level, the planner agent (p-agent) try to find the best locations for protecting resources in order to protect the important node. The lower level problem is represented as the shortest path problem in the network in which the edges are weighted with positive values and sometimes negative values. Thus, the Bellman-Ford algorithm is applied to solve the lower level problem. The NMPN is an NP-hard problem. In this work, the genetic, ant colony optimization, binary artificial bee colony with differential evolution, artificial bee colony algorithms, and a modified tabu search (MTS) algorithm are used to solve it. A test problem is randomly generated to investigate the performance of the

Received: July 2020; Accepted: March 2021
[*]Corresponding Author

metaheuristic algorithms in this paper. Parameters of the metaheuristic algorithms are tuned by the Taguchi method for solving the test problem. Also, the ANOVA test and Tukey's test are used to compare the performance of the metaheuristic algorithms. The best results are obtained by the MTS algorithm.

AMS Subject Classification: 90C10; 90B80; 68T20
Keywords and Phrases: Competitive facility location problem, Bilevel programming, Meta-heuristic algorithms

1 Introduction

Important nodes in a network should be protected. Usually, huge budgets are being taken to protect these nodes. For this purpose, a p-agent tries to find the best locations (or nodes) for placing protecting facilities. An appropriate location of protecting facilities may increase the resilience ability to protect the important node (which is called target node here) against potential threats such as natural disasters, deliberate attacks, etc. The location of protecting facilities must be determined with respect to the possible threats and existing restrictions on the resources and budgets. Against, a t-agent plans to access the target node. Usually, the more important achieving target node is for the t-agent, the more budgets and facilities are devoted by the t-agent for this object. One of the t-agent's plannings (tactics) to achieve the target node could be that the t-agent considers appropriate and secure places to reinforce itself during its movement towards the target. So, the p-agent must plan to protect the target node with respect to the t-agent's strategies for achieving the target node.

Facility location theory is one of the issues that is widely used in everyday life. In facility location problems, the objective is to determine the location of a set of facilities to minimize the cost of a set of demands with respect to some set of constraints. Usually, facility location problems are formulated for determining the optimal location of warehouses, distribution centers, hospitals, post offices, etc. The study of location theory was formally begun by Weber [30]. Weber located a warehouse so that the total distance of customers from this warehouse was minimized. This issue was not much considered until 1983. In 1983, the study of location theory was resumed with the publication of the article by Hakimi [12]. This paper was aimed to locate switch centers in a communication network and police stations on a highway. Afterward, with respect to the importance of location and allocating (resources allocation) resources correctly in saving costs and improving the efficiency in service delivery, numerous researches have been done in this regard [2, 11, 10, 13, 16, 27, 32].

Many research types about facility location problem have been done with this assumption that there is a monopoly in an environment between facilities [31]. But, another type of location problem, in which a decision-maker locates facilities based on the location of other competitive facilities, is called the competitive facility location (CFL) problem [31]. In other words, in the CFL problem, there is competition instead of monopoly between facilities. In such competitive environments, the rivals compete with each other to gain more share of the market. Nowadays, due to the lack of monopoly in the production of a product by a manufacturer or service delivery by an organization, studying competitive location models are more important than before.

Different classifications of the CFL problems have been proposed from different viewpoints. For example, Kress and Pesch divided the CFL problems based on the theoretical aspects of the game into static and dynamic categories [20]. Also, Revelle and Eiselt classified the CFL problems based on the location space into the d-dimensional and network classes [22]. Given that, an optimal location of all facilities is an equilibrium state in the CFL problems, and a categorization of the CFL problems is based on the type of equilibrium state, equilibrium state for the CFL problems is described as Nash equilibrium or Stackelberg equilibrium. The CFL problems based on Nash equilibrium are investigated by numerous researchers such as Hotelling and Eaton [9, 15]. On the other hand, the CFL problems based on Stackelberg equilibrium have been studied by multiple researchers such as Hakimi, Dreznzer, and Karkazis [8, 12, 18]. In this case, the CFL problems are formulated as a bi-level programming problem to obtain a Stackelberg equilibrium point.

In a bi-level programming problem, two decision decision-makers exist and decide sequentially. The decision-maker in the first level and the decision-maker in the second level are called leader and follower, respectively. In a bi-level programming problem, at first leader and then follower specify their strategy based on their profit. For a good introduction of the bi-level programming problem, we refer the readers to [6].

In 2007, a defensive location problem (DLP) was introduced by Uno and Katagiri, which is placed in the class of CFL problems [28]. In this problem, the defender wants to locate defensive facilities in some of the predetermined sites in order to prevent aggressors from reaching a strategic site. This strategic site is called core. In contrast, the invader looks for a proper path to achieve the core with respect to the specified strategy by the defender. They formulated the DLP as a bi-level programming problem. Given the fact that the defender may defend from several cores against the invader, thus they formulated a multi-objective defensive location problem.

In 2009, Berman investigated a defensive maximal covering problem on a network [3]. In this paper, it is assumed that a decision-maker (leader) wants to locate p facilities on the nodes of a network in order to provide maximum coverage of demands at the nodes of the network. Also, the decision-maker selects the facilities sites with this assumption that one of the network links will become unusable. In this problem, the leader's objective is to cover most demands after the elimination of a link. In contrast, the follower's objective is the elimination of the most damaging link.

In 2011, Uno and Kato formulated a multi-objective stochastic defensive location problem [29]. In this problem, the site and energy of the invader are considered as random variables. They used an interactive fuzzy satisfying method with a tabu search algorithm to obtain a satisfying solution for their problem.

In 2015, Khanduzi et al. formulated the continuous single-objective defensive location problem (CDLP) and proposed a hybrid tabu search algorithm to solve it [19]. In this problem, a decision-maker wants to locate different kinds of defensive facilities on nodes of a network to prevent the invader from reaching the core. Also, they solved it with a hybrid algorithm of the tabu search algorithm and Levenberg-Marquardt algorithm.

In 2016, Maleki et al. proposed a novel hybrid algorithm for solving

CDLP [21]. Their proposed hybrid algorithm integrates the imperialist competitive algorithm and Boyden-Fletcher-Golden-Shano (BFGS) algorithm. They have shown that the hybrid algorithm can find high-quality solutions for the CDLP problems.

In this paper, a new model based on DLP is presented where the t-agent can reinforce itself in some nodes of the network. This model may be applied in various fields such as computer networks security, flood control in a river, etc. In the following as an example, the modeling of the flood control problem in a river is explained as the NMPN. Suppose that a dam has been constructed on a river. Thus, there exists the possibility of the flood and increasing water volume in the river due to the dam break or the emergency gates opening. In modeling this problem as the NMPN, the p-agent is river engineers, and the t-agent is the flood. Also, the target node is a town that is along the river. In this problem, the river engineers want to reduce the amount of entering water volume from the overflowing river to the town. They can control the flood in the river by creating flood diversion channels in candidate sites with respect to existing constraints. The initial power of the t-agent can be equal to the amount of water volume that is entered into the river by the flood. The water volume of the overflowing river can be decreased due to water storage in the river and water guidance to the flood diversion channels. Also, besides the entry of a subsidiary branch into the river or the entry of floods of the surrounding catchment areas increases the volume of the river water, which can be considered as a place for the power reinforcement of the t-agent.

In NMPN, the p-agent intends to keep the t-agent away from the target node as much as possible. Thus, the p-agent locates protecting facilities with specified protecting capacity on some of the candidate sites with respect to the t-agent strategies and the existing constraints. In contrast, the t-agent exists in a specific site with some power. The t-agent wants to achieve the target node and needs the power to move on the network. Also, the t-agent's power is reduced in dealing with the p-agent's protecting facilities. Therefore, it can move towards the target node until it has power. In NMPN, the t-agent approaches the target node as much as possible, even by power reinforcement in its path toward the target node. In other words, the t-agent reinforces its power at the specified site provided that it can reach the target node or closer node to the target with power reinforcement. It is assumed that the p-agent locates protecting facilities on the nodes of a network. The t-agent's site, the target node, and the power reinforcement site of the t-agent are considered as the nodes of the network. The t-agent must find a proper path towards the target node on the network where weights of edges have positive and sometimes negative values.

Since DLP is a non-deterministic polynomial (NP)-time-hard problem, thus NMPN is NP-hard too [28]. Therefore, we can use metaheuristic algorithms to obtain the proper solutions for this problem. Metaheuristic algorithms are typically used for generating high-quality solutions in a proper time. In this paper, the genetic algorithm(GA), the ant colony optimization (ACO) algorithm, the binary artificial bee colony with differential evolution (BABC-DE) algorithm, the artificial bee colony (ABC) algorithm, and the MTS algorithm are used for solving the NMPN.

The innovations in this paper are as follows: The NMPN has been stated and modeled. The ABC, BABC-DE, ACO, MTS, and genetic algorithms are used for solving NMPN. A new method is proposed for obtaining heuristic information of the ACO algorithm. A new evaluation function is proposed that is applied in decreasing steps of protecting facilities of the tabu search algorithm. Moreover, the Taguchi method is used to perform the parameter tuning of the mentioned algorithms.

The rest of this paper is organized as follows: The NMPN is formulated as a bi-level programming problem in section 2. In section 3, solution approaches for solving the NMPN are described in details. Section 4 is devoted to the introduction of the Taguchi method. A computational experiment is performed in section 5. Finally, conclusions are given in Section 6.

2 Formulation of NMPN

The NMPN can be stated as a problem to determine whether a protecting facility should be located on a node of network or not when a t-agent with finite power wants to approach the target node as much as possible. In this section, the NMPN is formulated as a bi-level programming problem. In this modeling, it is assumed that the capacity of the t-agent to store power is α", there exists α' units of power at a specific site to reinforce t-agent's power, and $\alpha' \leq \alpha$". The t-agent must pay the cost to reinforce its power at the power reinforcement place. Since power reinforcement of the t-agent at the place of the power reinforcement needs to purchase and set up equipment. Hence, the t-agent reinforces its power at the specific site provided that it can reach the target node or closer node to the target. This cost can be used for purchasing and setting up the needed equipment to reinforce the power of the t-agent at the place of the power reinforcement.
In the following, the notations and assumptions applied in the formulation of NMNP are stated, and then the mathematical model of NMNP is presented.
Parameters
G: Network that the p-agent must locate its protecting facilities on it G= $(\boldsymbol{V},\boldsymbol{E})$.
$\boldsymbol{V}$: Set which contains nodes of the network. This set is equal to $\{v_1, ..., v_n\}$.
$\boldsymbol{E}$: Set which contains the edges of the network G and $|\boldsymbol{E}| = r$.
t: The node of G that the p-agent must protect from it against the t-agent.
v_e: The node of G that the t-agent reinforces its power on it.
ξ: The node of G which the t-agent is on it (the nth node of G).
e_{ij}: The connector edge between two nodes v_i and v_j of the network G.
w_{ij}: Weight of edge e_{ij} that is a positive value.
$\boldsymbol{p}_{vt}$: Set of all paths from the node v to the target node t.
$e^l(\boldsymbol{p})$: The lth edge in path $\boldsymbol{p}$.
$v^\lambda(\boldsymbol{p})$: The λth node from selected path $\boldsymbol{p}$ by the t-agent.
$\alpha(v^\lambda(\boldsymbol{p})|\boldsymbol{q})$: The power of the t-agent in the λth node of the path $\boldsymbol{p}$.
$\bar{\alpha}$: The initial power of the t-agent.
α'': The capacity of the t-agent for power storage.

α': Amount of power that is provided to reinforce the t-agent's power on node v_e.
β_1: The capacity of each protecting facility.
$\bar{c}$: The power reinforcement cost of the t-agent on node v_e.

Decision vector

$\boldsymbol{q}$: Vector $\boldsymbol{q} = (q_1, ..., q_{e-1}, q_{e+1}, ..., q_{n-1})$ is the decision vector of the p-agent. If there is no protecting facility on node i, then $q_i = 0$. Otherwise, $q_i = 1$.
$\boldsymbol{p}$: Vector $\boldsymbol{p}$ is the decision vector of the t-agent. This vector determines the path that the t-agent selects to reach t or the closest node to t as much as possible. In other words, the λth component of vector $\boldsymbol{p}$ represents the λth node in the path $\boldsymbol{p}$.
y_e: This variable is the decision variable of the t-agent. The value of this variable shows whether the t-agent reinforces itself on node v_e or not. If the t-agent reinforces its power on node v_e, then $y_e = 1$, otherwise $y_e = 0$.

Assumption

• The distance from v to the target node t, $d^t(v)$, is defined by the summation of the weights of edges in the shortest path from node v to the target node t. In other words

$$d^t(v) = \min_{\boldsymbol{p} \in \boldsymbol{P}_{vt}} \sum_{e_{ij} \in \boldsymbol{p}} w_{ij}$$

where $\boldsymbol{P}_{vt}$ is a set of all paths from the node v to the target node t.
• It is impossible to put protecting facility on nodes ξ and v_e.
• The initial power of the t-agent is $\bar{\alpha}$ before leaving ξ.
• The amount of power available to reinforce the t-agent's power on node v_e is α' units.
• The t-agent consumes its power in the two following ways:

1. By moving t-agent on the edge e_{ij}, its power is decreased by the amount w_{ij}.

2. when the t-agent passes the edge e_{ij} ($j \neq e$), its power is reduced by β_1 in case that a protecting facility was located on node v_j.

Also, if the t-agent moves on the edge e_{ie}, the t-agent's power will be increased by β_2 units, which β_2 is the amount of power that the t-agent can obtain on node v_e.
Therefore, if the t-agent moves from v_i to v_k , then the power of the t-agent is reduced as follows:

$$\bar{w}(e_{ik}, q) = \begin{cases} w_{ik} + \beta_1 q_k & k \neq e \\ w_{ie} - \beta_2 & k = e \end{cases}$$

where β_2 is calculated as follows:

$$\beta_2 = \min\{\alpha', \alpha'' - (\bar{\alpha} - \sum_{l=1}^{\gamma} \bar{w}(e^l(p), q))\}$$

• The t-agent can be in λth node of path $\boldsymbol{p}$, if the t-agent's power in λth node of path $\boldsymbol{p}$ is no negative, i.e.

$$\alpha(v^\lambda(\boldsymbol{p})|\boldsymbol{q}) = \bar{\alpha} - \sum_{l=1}^{\lambda} \bar{w}(e^l(\boldsymbol{p}), \boldsymbol{q}) \geq 0.$$

• When the t-agent's power reaches zero, the t-agent dies.

Mathematical formulation
Because the t-agent wants to approach the target node as much as possible; Thus the t-agent's objective function is defined as follows:

$$f^I(\boldsymbol{q}, \boldsymbol{p}) = \min\{d^t(v)|\ \alpha(v|\boldsymbol{q}) \geq 0\}$$

Also, if the t-agent can reach the closer node to the target node with power reinforcement, then it reinforces its power on node V_e. In other words, the t-agent pays the cost of the power reinforcement to reach the closer node to the target node. Thus, another objective function of the t-agent $\bar{c}(y_e)$ is as follows:

$$\bar{c}(y_e) = \bar{c} y_e$$

If the objective function of the p-agent is displayed with $f^d(\boldsymbol{q}, \boldsymbol{p})$, then $f^d(\boldsymbol{q}, \boldsymbol{p})$ satisfies in the following relation:

$$f^d(\boldsymbol{q}, \boldsymbol{p}) + f^I(\boldsymbol{q}, \boldsymbol{p}) = 0$$

In the following, the p-agent's feasible set is formulated. The constraints of the p-agent can be the cost of building and setting up facilities, staff allocation, budget restrictions, and so on. In this problem, we assume that the constraints of the p-agent are linear. If $\boldsymbol{A} \in \mathbb{R}^{m\times n-2}$ and $\boldsymbol{b} \in \mathbb{R}^{m\times 1}$ denote the coefficients matrix and the right-hand side vector, respectively, then the feasible set of the p-agent is defined as follows:

$$\boldsymbol{FD} = \{\boldsymbol{q} = (q_1, ..., q_{e-1}, q_{e+1}, ..., q_{n-1}) | \boldsymbol{Aq} \le \boldsymbol{b}\}$$

where elements of matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ are no negative. The feasible solutions set of the t-agent includes all paths from ξ to t that is shown by $\boldsymbol{SI}$. Therefore, the general form of the NMPN is as follows:

$$\begin{array}{ll} \max\limits_{\boldsymbol{q}} & f^d(\boldsymbol{q}, \boldsymbol{p}) \\ \text{where } \boldsymbol{p} \text{ solves} & \\ \min\limits_{y_e} & \bar{c}(y_e) \\ \min\limits_{\boldsymbol{p}} & f^d(\boldsymbol{q}, \boldsymbol{p}) \\ \text{s.t.} & \boldsymbol{q} \in \boldsymbol{FD}, \ \boldsymbol{p} \in \boldsymbol{SI} \end{array}$$

The t-agent wants to minimize two objective functions $\bar{c}(y_e)$ and $f^d(\boldsymbol{q}, \boldsymbol{p})$. The lexicographic method is an appropriate method for solving the second level problem, Since the minimization of the objective function $f^d(\boldsymbol{q}, \boldsymbol{p})$ is preferred to the minimization of the objective function $\bar{c}(y_e)$ by the t-agent.

Here, the structure of NMPN is illustrated with an example. Consider a network with 8 nodes and 10 edges. Suppose node 1 is the target node, the t-agent is on node 8, and the t-agent can reinforce its power on node 4. This network is represented in Fig 1. Also, suppose that $\alpha'' = \bar{\alpha} = 15$, $\alpha' = 7, \beta_1 = 2, A = [a_{11}, a_{12}, a_{13}, a_{15}, a_{16}, a_{17}] = [1, 1, 1, 1, 1, 1]$ and $b = 6$. If p-agent's decision vector be $\boldsymbol{q} = (q_1, q_2, q_3, q_5, q_6, q_7) = (0, 0, 1, 0, 1, 1)$, then the t-agent reaches the target node by using power reinforcement on node 4 and the t-agent must pay cost of power reinforcement on node 4. Also, path $8 \to 7 \to 4 \to 2 \to 1$ is selected to reach node 1 by the t-agent. But, if the t-agent on node 4 can not reinforce its power, then the t-agent stops on node 2, which is 10 units away from the target (node 1).

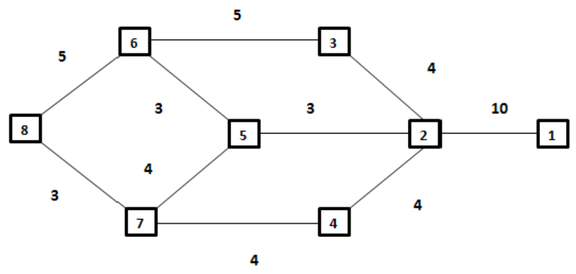


Figure 1: network with 8 nodes and 10 edges

3 The methodology of solving the NMPN

With regard to the presence of a power reinforcement node, the network of the NMPN includes edges with positive and sometimes negative weights. The objective function value, $f^d(\boldsymbol{q}, \boldsymbol{p})$, for given location $\boldsymbol{q} \in \boldsymbol{FD}$ is computed through finding the following:
1- Shortest paths from ξ to all nodes for location $\boldsymbol{q}$ and
2- Shortest paths from all nodes to t without considering the possibility of the t-agent's power reinforcement when no protecting facilities are located.
The Dijkstra algorithm finds the shortest paths from a source node to other nodes in a network that the weights of the edges are positive [5]. The complexity of the Dijkstra algorithm is $\mathbf{O}(r + nlogn)$. Thus, the Dijkstra algorithm is used to find shortest paths from all nodes to t. The Bellman-Ford algorithm is usually used to find the shortest path from a source node to other nodes of the network, when network edges are weighted with negative and positive values [5]. The computational complexity of the Bellman-Ford algorithm is equal to $\mathbf{O}(rn)$. Hence, the Bellman-Ford algorithm determines the shortest paths from ξ to all nodes for location $\boldsymbol{q}$. The NMPN is an NP-hard problem. Hence, the metaheuristic algorithms are a proper choice to solve this problem. Constraints handling is an important topic that must be considered in solving optimization problems with metaheuristic algorithms. Usually, a repairing strategy can be applied where search operators used in metaheuristic algorithms generate infeasible solutions.

In this paper, the GA, the ACO algorithm, the BABC-DE algorithm, the ABC algorithm, and the MTS algorithm are used for solving the NMPN. The repairing strategy used in these algorithms is proposed by Sakawa. For more information, readers are referred to [23]. In the following, the application of these algorithms to solve the NMPN is explained briefly.

3.1 Genetic algorithm

The GA is a metaheuristic algorithm that has been applied for solving many optimization problems. The GA has been developed by John Holland in the 1970s [14]. Each solution is represented by a chromosome in GA. Each chromosome consists of several genes. Usually, each gene represents the amount of a variable. The selection, crossover, and mutation operators are the main operators of GA which are used to produce a new population. In the crossover, two chromosomes (is called parents) exchange some of their parts in order to produce a pair of new chromosomes. The Mutation operator is applied to create unexpected changes in the value of some genes. In the following, the GA is investigated to solve the NMPN.

Initial population

The GA starts by generating an initial population of solutions. Various methods have been suggested for generating the initial population that one of these methods is the generation of the initial population of solutions, randomly. In this study, the initial population is generated randomly in GA.

Selection

This operator determines which individuals of the population are chosen in order to participate in crossover and mutation operations and how many offsprings are produced by each individual of the population. "The better is an individual, the higher is its chance of being a parent." is a basic principle of selection methods. The roulette wheel is used in this paper as one of the implementation methods of the selection operator.

We refer readers to [26], for more information about the roulette wheel selection method.

Crossover

Crossover is a binary and sometimes n-ary operator that is applied in GA to transfer some characteristics of the two parents to their offsprings. A crossover point is selected randomly in the 1-point crossover. Then portions of two parents that lie beyond the crossover point are exchanged to generate two new offsprings. Suppose that two selected parents are as follows:

$$\begin{aligned}\text{parent}_1 &= [q_{m1}, ...q_{mi-1}, q_{mi}, q_{mi+1}, ..., q_{mn-1}]\\ \text{parent}_2 &= [q_{d1}, ...q_{di-1}, q_{di}, q_{di+1} ..., q_{dn-1}]\end{aligned}$$

Also, suppose that the ith component is selected as a crossover point. Thus the generated offsprings are as follows:

$$\begin{aligned}\text{offspring}_1 &= [q_{m1}, ...q_{mi-1}, q_{mi}, q_{di+1}, ..., q_{dn-1}]\\ \text{offspring}_2 &= [q_{d1}, ...q_{di-1}, q_{di}, q_{mi+1} ..., q_{mn-1}]\end{aligned}$$

In the GA, the production rate of new offsprings by the crossover operator is called the crossover rate and is denoted by p_c. In other words, if the population size is shown with N_{pop}, then $p_c \times N_{pop}$ new offsprings should be produced by using the crossover operator.

Mutation

The mutation is a unary operator. In other words, it is applied to a single individual of the population. The rate of production of offsprings by mutation operator is called mutation rate, which is represented by p_m. Therefore, the number of generated offsprings by mutation operator is $p_m \times N_{pop}$. Uniform mutation as the simplest mutation operator is performed by choosing a gene randomly, and then the value of this gene is changed by the flip operator.

Next generation

The replacement strategy is used to decide which individuals remain in the population and which individuals are replaced with new offsprings. One of the replacement strategies is the elitism strategy that is performed by selecting the best individuals between the parents and the offsprings in order to generate a better population.

Stopping criteria

In this paper, iterations number is the stopping criteria of the algorithm.

The implementation of the GA to solve the NMPN is as follows:
Step 0: Determine appropriate values of crossover rate (p_c), mutation rate (p_m), size of the population (N_{pop}), number of iterations (IT), and input parameters of the NMPN.
Step 1: Generate an initial population of solutions with size N_{pop}.
Step 2: Apply repair operator for the infeasible solutions of the initial population and evaluate all feasible solutions.
Step 3: Generate $p_c \times N_{pop}$ new solutions by the crossover operator.
Step 4: Apply repair operator for the infeasible solutions generated by the crossover operator and then evaluate all feasible solutions.
Step 5: Generate $p_m \times N_{pop}$ new solutions by mutation operator.
Step 6: Apply the repair operator for the infeasible solutions generated by the mutation operator and evaluate all feasible solutions.
Step 7: Add a unit to the counter of iterations.
Step 8: Apply the elitism strategy to select appropriate individuals who are survived.
Step 9: Repeat steps 3-8 until the maximum number of iterations is established.

3.2 Ant colony optimization algorithm

Dorigo developed an ant colony optimization (ACO) algorithm in 1997 [7]. The ACO algorithm is a population based meta-heuristic algorithm that mimics the foraging behavior of ants in their colony. Every ant leaves a trail of pheromone on his movement path that evaporates quickly. But it remains as a track in a short time. Each ant chooses a

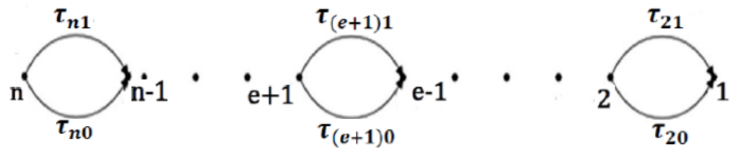


Figure 2: Routing Diagram for Ants in the NMPN

path from the nest towards the food source with respect to the concentration amount of pheromone on the paths.

The ACO algorithm is inspired by the interaction between ants in a colony to find the shortest path from the nest towards the food source. In nature, a colony of ants may find the shortest path from the nest towards a food source through the pheromone concentration amount on the paths. The ACO algorithm is an iterative algorithm for solving combinatorial optimization problems. At each iteration of the algorithm, every ant constructs a solution, and then the pheromone of the selected paths will be updated. In the following, we use the binary ACO algorithm to solve the NMPN.

Ant colony optimization algorithm for solving NMPN

In this algorithm to solve the NMPN, an ant constructs a solution by stepping on the described routing graph in Fig 2. In other words, the construction of a solution in the ACO algorithm is obtained by the ant's movement from the nth node to the first node on the routing graph. Also, every ant moves from node i to node $i-1$ through the upper path i_0 or the lower path i_1. If an ant chooses path i_0, it means that there is no protecting facility on the ith node of the network. Also, if an ant chooses path i_1, it means that a protecting facility is located on the ith node of the network. In other words, an ant selects a path, but its result is the location of the protecting facilities on the nodes of the network of NMPN. At the first, the initial value of pheromone is determined on paths i_0 and i_1. Afterward, two phases of solution construction and pheromone update are repeated. When all ants find a solution, then each ant reports the quality of its solution through depositing pheromone on its movement path. Suppose that the initial pheromone value on paths i_0 and i_1 of the routing graph is determined as follows:

$$\tau_{i0}^0 = \frac{\gamma}{n_i}, \quad \tau_{i1}^0 = \frac{\gamma}{n_i}, \; i = 2, ..., n-1, \; i \neq e$$
$$\tau_{10}^0 = \gamma, \quad \tau_{11}^0 = \gamma$$

where γ is a constant coefficient and n_i is the number of output edges of node v_i in the network.

Solution Construction

Suppose that $\boldsymbol{q'} = (q'_1, q'_2, ...q'_{e-1}, q'_{e+1}, ..., q'_{n-1})$ is a solution of the NMPN which is constructed by an ant in the ACO algorithm. An ant determines the value of the component q'_i of solution vector $\boldsymbol{q'}$ by choosing one of the two available paths for movement from node i to node $i-1$. Ants use a probability law to choose one of two paths i_0 and i_1, on the routing graph in Fig 2. This law is determined through heuristic and pheromone information of the paths i_0 and i_1. The heuristic information about choosing the path i_1 can be found as follows:

$$\eta_{i1} = \frac{\max\limits_i d^t(v_i) - d^t(v_i)}{\max\limits_i d^t(v_i) - \min\limits_i d^t(v_i)}$$

Also, heuristic information of path i_0 can be obtained as follows:

$$\eta_{i0} = \frac{\sum\limits_{j=1}^{m} a_{ij} - \min\limits_i \sum\limits_{j=1}^{m} a_{ij}}{\max\limits_i \sum\limits_{j=1}^{m} a_{ij} - \min\limits_i \sum\limits_{j=1}^{m} a_{ij}}$$

The pheromone information of path $i_1(i_0)$ i.e. $\tau_{i1}(\tau_{i0})$ is a criterion to locate(not to locate) protecting facility on the ith node of the network based on the history of the movement in the past. The path i_0 is selected b kth ant with probability:

$$p_{i0}^k = \frac{\tau_{i0}^{\alpha}\eta_{i0}^{\beta}}{\sum\limits_{j=0}^{1} \tau_{ij}^{\alpha}\eta_{ij}^{\beta}} \tag{1}$$

The kth ant selects path i_1 with probability:

$$p_{i1}^{k} = \frac{\tau_{i1}^{\alpha}\eta_{i1}^{\beta}}{\sum\limits_{j=0}^{1}\tau_{ij}^{\alpha}\eta_{ij}^{\beta}} \tag{2}$$

where α and β are two parameters that represent the relative importance of the pheromone and heuristic information, respectively.

Pheromone Update

Suppose that $\boldsymbol{q} = (q_1, ..., q_{e-1}, q_{e+1}, ..., q_{n-1})$ is a solution of the NMPN. The pheromone update rule for paths i_0 and i_1 is as follows:
If $q_i = 1$, then pheromone of paths i_0 and i_1 are updated as follows:

$$\tau_{i1}^{new} = \tau_{i1} + (1 - \frac{1}{f^d(\boldsymbol{q}, \boldsymbol{p})}),\ \tau_{i0}^{new} = \tau_{i0} \tag{3}$$

If $q_i = 0$, then pheromone of paths i_0 and i_1 are updated as follows:

$$\tau_{i0}^{new} = \tau_{i0} + (1 - \frac{1}{f^d(\boldsymbol{q}, \boldsymbol{p})}),\ \tau_{i1}^{new} = \tau_{i1} \tag{4}$$

Pheromone on the edges of the routing graph evaporate to search different paths of the graph by using the following formula:

$$\tau = (1 - \rho)\tau,\ \ \rho \in (0, 1] \tag{5}$$

where ρ is the evaporation coefficient.
The implementation of the ACO algorithm is as follows:
Step 0: Determine the number of ants in a colony(N_{ant}), pheromone evaporation rate (ρ), the relative importance of heuristic information (α), the relative importance of pheromone information(β), number of iterations (IT), and coefficient of initial pheromone value on routing graph edges (γ). Input parameters of the NMPN. Let the best solution equal to $-\infty$.
Step 1: Produce a solution for each ant in the colony by using (1) and (2).
Step 2: Apply repair operator for infeasible solutions.

Step 3: Evaluate feasible solutions.
Step 4: Update the pheromone of the selected paths by using (3) and (4).
Step 5: Evaporate the pheromone on the edges of the routing graph by using (5).
Step 6: Update the best solution.
Step 7: Repeat steps 1-6 until the maximum number of iteration is established.

3.3 Artificial bee colony algorithm

The artificial bee colony (ABC) algorithm is a swarm based approach that is proposed for optimizing continuous numerical functions in 2005 by Karoboga [17]. The ABC algorithm simulates the foraging behavior of honey bees in searching for food sources. There are three types of artificial bees in the ABC algorithm: employed, onlooker, and scout bees. Each food source is allocated to an employed bee which produces a modification on the position of her food source with respect to local information. If the nectar amount of the new food source is more than the previous one, the employed bee memorizes the new position and forgets the old one. When the employed bees come back to the hive, they share their information through waggle dance with onlooker bees. Each onlooker bee uses the information of the employed bees to select a food source probabilistically. An onlooker bee searches a new food source in the neighborhood of her selective food source in order to achieve a food source with more nectar. The scout bees fly in the search space randomly. Employed, onlooker, and scout bees are three iterative steps in each iteration of the ABC algorithm.

Artificial bee colony algorithm for solving NMPN

Here, the ABC algorithm is investigated to solve the NMPN. Half of a colony includes employed bees, and another half consists of the onlooker bees in the ABC algorithm. The number of employed bees is equal to the number of food sources. In the ABC algorithm, each food source and its nectar amount represents a solution and quality of the solution

in the optimization problems, respectively.
At first, SN position of food sources $\boldsymbol{q_i} = (q_{i1}, ..., q_{e-1}, q_{e-1}, ..., q_{n-1})$ are generated randomly in ABC algorithm by:

$$q_{ij} = q_j^{min} + \text{rand}(0,1)(q_j^{max} - q_j^{min}) \tag{6}$$

where q_j^{max} and q_j^{min} are lower and upper bounds of the jth component of solutions q_i, respectively. Each food source is allocated to an employed bee. We propose the following transfer function to shift the solution from continuous space to binary space.

$$q_{ij} = \begin{cases} 1 & \frac{\exp(q_{ij}) - \exp(-q_{ij})}{\exp(q_{ij}) + \exp(-q_{ij})} > 0 \\ 0 & \frac{\exp(q_{ij}) - \exp(-q_{ij})}{\exp(q_{ij}) + \exp(-q_{ij})} \leq 0 \end{cases} \tag{7}$$

Employed bee phase

Each employed bee searches for a new food source in the neighborhood of her food source in order to find a food source with more amount of nectar as follows:

$$\hat{q}_{ij} = q_{ij}(t) + \phi_{ij}(q_{ij}(t) - q_{kj}(t)), \tag{8}$$

where $\boldsymbol{q}_k(t)(k \in \{1, 2, ..., SN\},\ k \neq i)$ is a neighbor of $\boldsymbol{q}_i(t)$ selected randomly and ϕ_{ij} is a random number in $[-1, 1]$.

Onlooker bee phase

When all employed bees completed the search process, they share obtained information about the nectar amount of their food sources with onlooker bees through waggle dance. Then each onlooker bee chooses a food source by the roulette wheel. The probability of selecting ith food source is obtained as follows:

$$p_i = \frac{fit_i}{\sum\limits_{i=1}^{SN} fit_i} \tag{9}$$

where fit_i is the nectar amount (fitness) of the ith food source. Then, a food source can be generated in the neighborhood of the selected food source q_i as follows:

$$\tilde{q}_{ij} = q_{ij}(t) + \psi_{ij}(q_{ij}(t) - q_{kj}(t))$$

where $\boldsymbol{q}_k(t)(k \in \{1, 2, ..., SN\})$ is a neighbor of $\boldsymbol{q}_i(t)$ selected randomly and ψ_{ij} is a random number in the range of $[-1, 1]$. The transfer function (7) is used to convert the obtained solution to a solution in binary space.

Scout bee phase

One of the steps in the ABC algorithm is to abandon food sources without any improvements in their amount of nectar in a predetermined number of cycles and replace them with a new food source that is generated randomly. The value of the predetermined number of cycles for the abandonment of a food source is called the abandonment limit. The new food source is discovered by a scout bee as follows:

$$q_{ij} = q_j^{min} + \text{rand}(0, 1)(q_j^{max} - q_j^{min}) \tag{10}$$

The implementation of the ABC algorithm to solve the NMPN is followed by:
Step 0: Input parameters of the NMPN. Also, initialize the number of iterations (IT), population size of solutions (SN), and abandonment limit (AL).
Step 1: Generate an initial population of solutions q_{ij} , $i = 1, ..., SN,\ j = 1, ..., n-1,\ j \neq e$, by using (6).
Step 2: Apply repair operator for infeasible solutions.
Step 3: Evaluate the fitness value of the initial solutions $\boldsymbol{q}_i,\ i = 1, ..., SN$.
Step 4: Generate new solutions $\hat{q}_{ij},\ i = 1, ..., SN,\ j = 1, ..., n-1,\ j \neq e$ for the employed bees by using (8).
Step 5: Apply repair operator for infeasible solutions $\hat{\boldsymbol{q}}_i,\ i = 1, ..., SN$.
Step 6: Evaluate fitness value of $\hat{\boldsymbol{q}}_i,\ i = 1, ..., SN$.
Step 7: Use the greedy selection procedure between $\hat{\boldsymbol{q}}_i$ and $\boldsymbol{q}_i$.
Step 8: Calculate probability values $p_i,\ i = 1, ..., SN$, for the solutions $\boldsymbol{q}_i$ by using (9).
Step 9: Produce solutions $\tilde{\boldsymbol{q}}_i,\ i = 1, ..., SN$, from the selected solutions $\boldsymbol{q}_i$ by using values p_i and roulette wheel selection method.
Step 10: Apply repair operator for infeasible solutions $\tilde{\boldsymbol{q}}_i$.

Step 11: Evaluate fitness value of $\tilde{\boldsymbol{q}}_i$.
Step 12: Use the greedy selection procedure between $\tilde{\boldsymbol{q}}_i$ and $\boldsymbol{q}_i$.
Step 13: Specify abandoned solutions and replace them with new solutions generated randomly using (10).
Step 14: Update the best found solution.
Step 15: Repeat Steps $4 - 14$ until the convergence condition of the maximum number of iterations is established.

3.4 Binary artificial bee colony with differential evolution algorithm

A binary artificial bee colony with differential evolution (BABC-DE) algorithm has been designed for solving binary knapsack problem in 2009 by Cao et al. [4]. The differential evolution (DE) algorithm is proposed by Storn and Price in 1995 [24]. The BABC-DE algorithm is a hybridization of the artificial bee colony algorithm and differential evolution algorithm to obtain a more efficent algorithm than ABC and DE. The BABC-DE algorithm is a modified version of the ABC algorithm. Similar to the ABC algorithm employed, onlooker, and scout bees are three iterative phases in the BABC-DE algorithm. In the BABC-DE algorithm, mutation and crossover operators of the DE algorithm are used in the onlooker bee phase. Also, a binary operator is used in the employed bee phase of the BABC-DE algorithm [4]. In the design of this operator, the memory and neighbor information are considered simultaneously to increase the efficiency of the BABC-DE algorithm.

Artificial bee colony with differential evolution algorithm for solving NMPN

At first, SN food sources $\boldsymbol{q}_i = (q_{i1}, ..., q_{ie-1}, q_{ie+1}, ..., q_{in-1})$, $i = 1, ..., SN$, are generated randomly, in the BABC-DE algorithm. In this paper, initial food sources are generated as follows:

$$q_{ij} = \begin{cases} 0 & \phi_{ij} > 0 \\ 1 & \phi_{ij} \leq 0 \end{cases} \tag{11}$$

where ϕ_{ij} is a random number in the range of [0,1].

Employed bee phase

In the employed bee phase, solution $\hat{\boldsymbol{q}}_i$ is generated by solution $\boldsymbol{q}_i$ as following:

$$\hat{q}_{ij} = mod(q_{k_1 j} + |q_{ij}(t) - q_{k_2 j}(t)|, 2) \tag{12}$$

where function $mod(\cdot, 2)$ calculates the remainder of the division of the first part by 2, $k_1, k_2 \in \{1, ..., SN\}$ are selected randomly and are held in $k_1 \neq k_2 \neq i$.

Onlooker bee phase

In the BABC-DE algorithm, an onlooker bee chooses the ith food source with probability value $\hat{p}_i$ that is calculated as follows:

$$\hat{p}_i = \frac{fit_i}{\sum\limits_{i=1}^{SN} fit_i} \tag{13}$$

where fit_i is fitness of the ith food source. Then, the onlooker bee determines a neighbor food source around the chosen one by mutation and crossover operator. The proposed mutation operator by Cao is defined as follows:

$$\tilde{q}_i = \begin{cases} M(\boldsymbol{q}_{gb}(t)) & r < w \\ M(\boldsymbol{q}_k(t)) & r \geq w \end{cases} \tag{14}$$

where r is a random number within [0,1], integer number $k \neq i$ is selected randomly from set $\{1, ..., SN\}$. Operator $M(\cdot)$ is a binary swap mutation operator described in Table (1). The food source $\boldsymbol{q}_{gb}(t)$ is the best food source in the iteration t. Also, the selective coefficient w is used to control the frequency of mutation on food sources $\boldsymbol{q}_{gb}(t)$ and $\boldsymbol{q}_k(t)$ in order to prevent premature convergence. In iteration t, w can be obtained as follows:

Table 1: Binary mutation operator in BABC-DE

Step 1	select two integers,$u, v \in \{1, 2, ..., e-1, e+1, ..., n-1\}, u \neq v$, randomly.
Step 2	If $q_{iu}(t) = q_{iv}(t)$ then $q_{iu}(t) = ((q_{iu}(t) + 1)mod2)$
Step 3	If $q_{iu}(t) \neq x_{iv}(t)$ then $q_{iu}(t) = ((q_{iu}(t) + 1)mod2)$ and $q_{iv}(t) = ((q_{iv}(t) + 1)mod2)$

$$w = w_{start} - \frac{w_{start} - w_{end}}{IT} \times t$$

where IT is the number of iterations. After obtaining a mutated individual, a crossover individual is obtained by:

$$u_{ij} = \begin{cases} \tilde{q}_{ij} & r_j \leq CR \text{ } or \text{ } j = j_{\text{rand}} \\ q_{ij}(t) & \text{otherwise} \end{cases} \tag{15}$$

where r_j, $j = 1, ..., e-1, e+1, ..., n-1$, is a random number in the range of [0,1]. Integer number j_{rand}, $j_{rand} \neq e$, is generated randomly within $[1, e-1]$ or $[e+1, n-1]$ which is selected randomly. Also, CR is the crossover rate in the range of [0,1].

Scout bee phase

In the BABC-DE algorithm, the scout bee phase is designed to avoid trapping algorithm in local optimal. If the ith food source does n't improve in a certain number of iterations, then the food source $\boldsymbol{q}_i$ would replace with a new food source that is generated by (11).
Implementation of the BABC-DE algorithm is described by using the following steps:
Step 0: Initialize the population size of solutions (SN), number of iterations (IT), w_{start}, w_{end}, and abandonment limit (AL). Input parameters of the NMPN.
Step 1: Initialize SN solutions by using (11).
Step 2: Apply repair operator for infeasible solutions.
Step 3: Evaluate fitness value of each of the SN solutions.
Step 4: Find the best solution $\boldsymbol{q}_{gb}(t)$.
Step 5: Generate solution $\hat{\boldsymbol{q}}_i(t)$ using (12).
Step 6: Apply repair operator for $\hat{\boldsymbol{q}}_i(t)$, if $\hat{\boldsymbol{q}}_i(t)$ is infeasible .
Step 7: Evaluate fitness value of solution $\hat{\boldsymbol{q}}_i(t)$.
Step 8: Apply a greedy selection mechanism between $\boldsymbol{q}_i(t)$ and $\hat{\boldsymbol{q}}_i(t)$.
Step 9: Update $\boldsymbol{q}_{gb}(t)$, if fitness value of $\boldsymbol{q}_i(t)$ is more than fitness value of $\boldsymbol{q}_{gb}(t)$.
Step 10: Repeat SN times, steps $5 - 9$.
Step 11: Select solution $\boldsymbol{q}_i(t)$ probabilistically by using calculated values $\hat{p}_i$ through (13) and roulette wheel selection mechanism.
Step 12: Generate mutated solution $\tilde{\boldsymbol{q}}_i(t)$ by (14).
Step 13: Generate crossover solution $\boldsymbol{u}_i(t)$ by (15).
Step 14: Apply repair operator for solution $\boldsymbol{u}_i(t)$, if $\boldsymbol{u}_i(t)$ is infeasible.

Step 15: Evaluate fitness value of solution $\boldsymbol{u}_i(t)$.
Step 16: Apply a greedy selection mechanism between $\boldsymbol{q}_i(t)$ and $u_i(t)$.
Step 17: Update $\boldsymbol{q}_{gb}(t)$, if fitness value of $\boldsymbol{q}_i(t)$ is more than fitness value of $\boldsymbol{q}_{gb}(t)$.
Step 18: Repeat SN times, steps $11 - 17$.
Step 19: Generate a new solution by using (11) for replacing with solution $\boldsymbol{q}_i(t)$, if $\boldsymbol{q}_i(t)$ has not been improved in AL trials.
Step 20: Let $t = t + 1$.
Step 21: Repeat Steps $4 - 20$ until the convergence condition of the maximum number of iterations is established.

3.5 Modified tabu search algorithm

Here, a modified tabu search algorithm is investigated to solve the NMPN. Assume that the set $\boldsymbol{S}_M$ is defined as follows:

$$\boldsymbol{S}_M = \{\pm \boldsymbol{t}^i |\ i = 1, ..., e-1, e+1, ..., n-1\},$$

where $\boldsymbol{t}^1 = (1, ..., 0), ...,$ and $\boldsymbol{t}^{n-1} = (0, ..., 1)$. Each member of $\boldsymbol{S}_M$ is called move. Assume that, set of neighbors $\boldsymbol{q}$ is defined as follows:

$$N(\boldsymbol{q}) = \boldsymbol{q} + \boldsymbol{S}_M$$

Let tabu list $\boldsymbol{T}$ be a set with a specific size that consists of recent chosen moves. No members of $\boldsymbol{T}$ can be selected until specific number of iterations are done. In the tabu search algorithm, a non-tabu move is chosen based on the improvement amount of an evaluation function, which is usually the objective function of an optimization problem. The tabu search algorithm by Uno and Katagiri is constructed based on two types of steps: protecting facilities increment and decrement [28]. We propose a new evaluation function in case of facilities decrement in the Uno's tabu search algorithm. If q is feasible, when the number of located protecting facilities is decreased, then evaluation function (EF) is defined as:

$$EF = \max_j \{\boldsymbol{e}\boldsymbol{A}_j |\ q_j = 1,\ \ -t^j \notin T,\ j = 1, .., e-1, e+1, ..., n-1\}$$

where dimensional of vector $\boldsymbol{e}$ is $1 \times m$, and all of its components are 1. If $j < e$, the vector A_j is the jth column of matrix A, and if $j > e$, the

vector A_j is the $(j-1)$th column of matrix A.
Otherwise, the evaluation function (EF) is defined as:

$$EF = \max_j \{ \sum_{i \in I} a_{ij} |\ q_j = 1,\ \ -t^j \notin T,\ j = 1, .., e-1, e+1, ..., n-1 \}$$

where the set $\boldsymbol{I}$ is defined as follows:

$$\boldsymbol{I} = \{ i | \sum_{j=1, j \neq e}^{n-1} a_{ij} q_j > b_i,\ i = 1, ..., m \}$$

In this paper, Uno's tabu search algorithm is considered with a new evaluation function which is used in steps of facilities decrement.

4 Taguchi method

Values (levels) of parameters affect the quality of the solutions and the run time of metaheuristic algorithms. Thus appropriate levels should be determined for these parameters by using parameter tuning methods. Assume that a metaheuristic algorithm has n controllable parameters, and each parameter has k levels. A full factorial design needs n^k experiments. When all experiments have been conducted, the "best" level can be determined for each parameter. Hence, this approach has a high computational cost. One of the most popular methods for parameter tuning is the Taguchi method. The Taguchi method is introduced by Taguchi [25]. In the Taguchi method, parameters are divided into two categories: controllable and uncontrollable parameters. The value of controllable parameters can be specified and controlled easily, whereas uncontrollable parameters are difficult to control in normal situations. In the Taguchi method, the number of experiments is decreased by using orthogonal arrays. The freedom degree of each parameter in a metaheuristic algorithm is considered as the number of levels minus one. In the Taguchi method, the sum of the freedom degrees of parameters of an algorithm plus one is the minimum number of required rows in an appropriate orthogonal array. When the experiments have been performed, the Taguchi method uses a statistical measure to evaluate the performance of algorithms that is called signal-to-noise (S/N) ratio. The terms “signal” and “noise” indicate the desirable (mean response variable) and undesirable values (standard deviation), respectively. For a minimization problem, the S/N ratio is calculated as follows:

$$\frac{S}{N} = -10 \log_{10}^{y^2}$$

where y is the objective function value. The process of parameter tuning with the highest S/N ratio yields better results because it guarantees optimum quality with minimum variance [1]. Thus, the best level for each parameter in a metaheuristic algorithm corresponds to the level that has the highest value of the S/N ratio.

It is better to unscale values of the objective function in each level of the suggested experiments by the Taguchi method. The objective function values can be unscaled by using the related percentage deviation (RPD) formula. The values of RPD is obtained as follows:

$$RPD = \frac{|\text{Current Solution} - \text{Best Solution}|}{\text{Best Solution}} \times 100$$

where ” Best Solution” is the best value of the objective function in all experiments for a metaheuristic algorithm. Then, the marginal mean of S/N ratio values for each level of the parameters are computed, and the best level for each parameter is specified by using obtained mean values.

5 Computational experiments

The purpose of a computational experiment is for the validation of the model. Thus a numerical example was generated to illustrate the NMPN randomly. This example is solved by five metaheuristic algorithms which are the genetic, ACO, ABC, BABC-DE, and MTS algorithms. The proposed algorithms are coded in MATLAB and run on a machine with an Intel Core i7 , 3.60 GHz CPU, and 32 GB RAM.

Random generation of a numerical example

Consider a network with n=200 nodes and 985 edges. The weights of edges in this network are selected from set $\{1, ..., 30\}$, randomly. The t-agent's initial power amount is equal to $\bar{\alpha} = 200$. Moreover, it is assumed that $\alpha^{"} = \bar{\alpha}$. The capacity of each protecting facility is equal to 50. In this example, node v_e is the 94th node of the network which was selected randomly. Assume that the maximum amount of power that a t-agent can obtain at this node is equal to 70 units. For all $i = 1, ..., m$ and $j = 1, ..., n-1, j \neq e$, the entry a_{ij} of coefficients matrix $A^{m \times n-2}$ is a randomly selected scalar in set $\{1, .., 30\}$. The elements of the right-hand side vector $\boldsymbol{b}$ are determined as follows:

$$b_i = \theta_i \sum_{j=1,\ j\neq e}^{n-1} a_{ij}, \quad i = 1, ..., m,$$

where θ_i is a random value in interval $[0.1, 0.2]$.

parameter tuning

The parameter tuning and assigning an appropriate level for each of the parameters influence the solution quality and run time of metaheuristic algorithms. In this paper, parameter values are determined by using the Taguchi method for each algorithm. Five levels are considered for each of the parameters in each algorithm. Taguchi method suggests 25 (L_{25}) orthogonal arrays for each algorithm. Also, for each algorithm, each level of suggested experiments are performed three times in order to obtain more reliable information.

Controllable parameters in GA are: probability for the crossover (pc), probability for the mutation (pm), population size ($Npop$), and number of iterations (IT). The mean of S/N ratios plot for each level of parameters of GA is shown in Fig 3. Also, tested values and suggested value for each parameter of the GA are presented in Table (2).

Parameters of ACO algorithm for solving the NMPN are: coefficient related to the initial value of pheromone on the edges of routing graph(γ), relative importance (weight) of heuristic information (α), relative importance (weight) of pheromone information (β), number of ants in each iteration of algorithm (N_{ant}), number of iterations (IT). There are considered five levels for each of these parameters. The best level of each parameter in the ACO algorithm is determined by the mean of

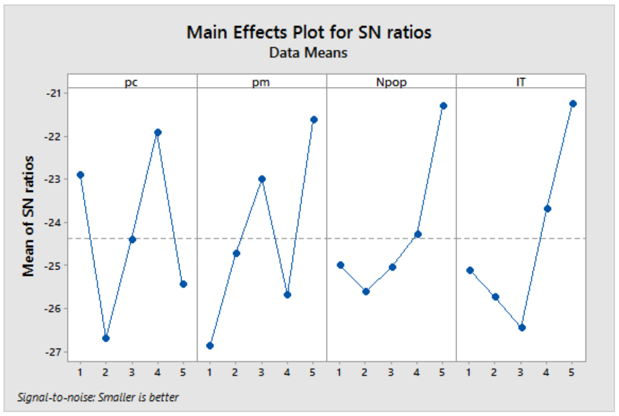


Figure 3: GA

Table 2: Parameter tuning results of GA for the numerical example

Parameter	Tested values	Suggested value
p_c	0.65,0.7,0.75,0.8,0.85	0 .8
p_m	0.15,0.2,0.25,0.3,0.35	0.35
N_{pop}	30,50,70,100,130	130
IT	100,140,180,220,260	260

S/N ratios. The mean of S/N ratios plot for each level of parameters in the ACO algorithm is shown in Fig 4. Also, tested values and suggested value for each parameter of the ACO algorithm is presented in Table (3).

In the ABC algorithm, controllable parameters are: number of iterations (IT), population size of solutions (SN), and abandonment limit (AL). To solve the numerical example of the NMPN by ABC, the abandonment limit is considered as follows:

$$AL = \text{round}(C \times SN \times (n - 2)) \tag{16}$$

where C is the constant value within $[0, \infty)$. So, according to the definition of AL, C and SN parameters must be tuned instead of parameter AL. The mean of S/N ratios plot for each level of the parameters in the ABC algorithm is shown in Fig 5. Also, the tested values and suggested

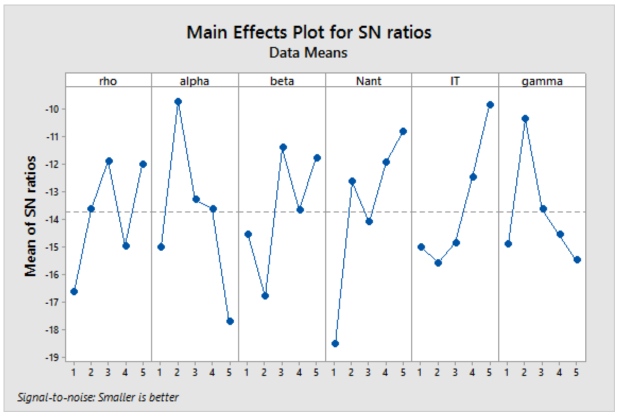


Figure 4: ACO algorithm

Table 3: Parameter tuning results of the ACO algorithm for solving the numerical example

Parameter	Tested values	Suggested value
ρ	0.01,0.0575,0.105,0.1525,0.2	0 .105
α	0 .5,1,1.5,2,2.5	1
β	0.5,1,1.5,2,2.5	1.5
N_{ant}	10,15,20,25,30	30
IT	100,140,180,220,260	260
γ	0.1,0.3,0.5,0.7,0.9	0.3

value of each parameter are presented in Table 4.

In the BABC-DE algorithm, adjustable parameters are: w_{start}, w_{end}, crossover rate (CR), number of iterations (IT), population size of solutions (SN), and abandonment limit (AL). In this study, the abandonment limit in the BABC-DE algorithm is calculated by (16). So, similar to the ABC algorithm, the parameter C and parameter SN must be tuned instead of the parameter AL. The mean of S/N ratios plot for each level of the parameters in the BABC-DE algorithm is shown in Fig 6. Tested values and suggested value of each parameter are presented in Table (5).

In the proposed MTS algorithm, controllable parameters are: tabu list length(TL), number of terminal delete moves (NTD), number of

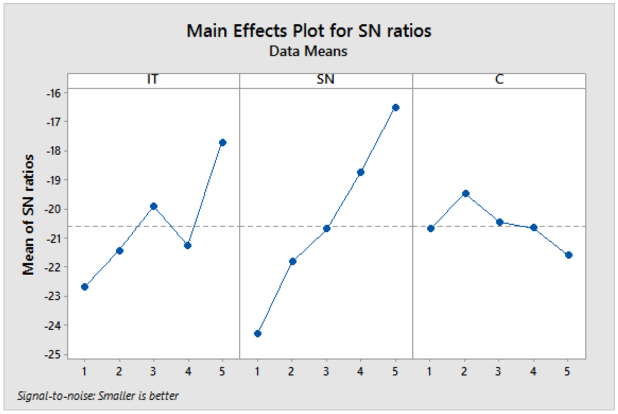


Figure 5: ABC algorithm

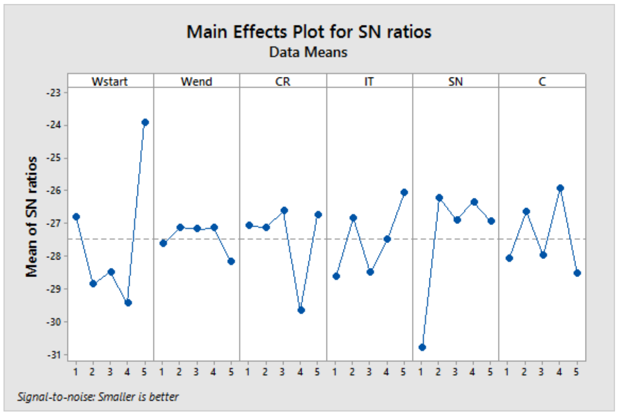


Figure 6: BABC-DE algorithm

Table 4: Parameter tuning results of the ABC algorithm for solving the numerical example

Parameter	Tested values	Suggested value
IT	100,140,180,220,260	260
SN	10,15,20,20,30	30
C	0.01,0.0575,0.0105,0.1525,0.2	0.0575

Table 5: Parameter tuning results of the BABC-DE algorithm for solving the numerical example

Parameter	Tested values	Suggested value
IT	100,140,180,220,260	260
CR	0.6,0.7,0.8,0.9,1	0.8
SN	10,15,20,25,30	15
w_{start}	0.2,0.3,0.4,0.5,0.6	0.6
w_{end}	0.6, 0.7,0.8,0.9,1	0.7
C	0.01,0.0575,0.105,0.1525,0.2	0.1525

iterations(IT). The mean of S/N ratios plot for each level of the parameters of the MTS algorithm is shown in Fig 7. Also, tested values and suggested value of each parameter are presented in Table (6).

Table 6: Parameter tuning results of the MTS algorithm for solving the numerical example

Parameter	Tested values	Suggested value
TL	20,30,40,50,60	50
NTD	5,6,7,8,9	8
IT	100,140,180,220,260	140

Statistical analysis

Statistical analysis is a powerful tool to evaluate the performance of metaheuristic algorithms. A common statistical analysis is the analysis of variance (ANOVA) test. In this paper, the ANOVA test is performed to investigate whether the difference between means of metaheuristic

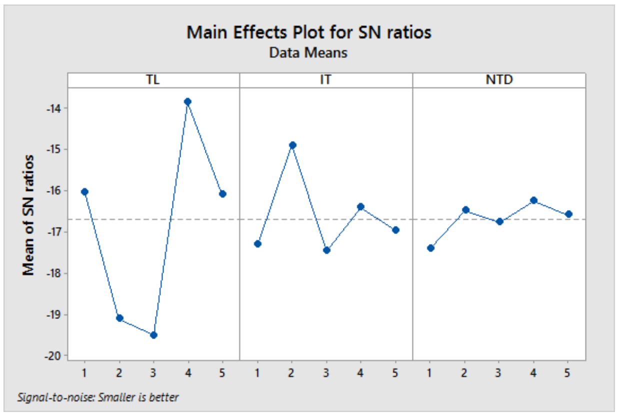


Figure 7: MTS algorithm

algorithms is statistically significant or not. If a difference exists somewhere between algorithms, the p-value associated with the ANOVA test indicates it. For small values of the p-value (< 0.05), the ANOVA test confirms that there is a statistically significant difference between the mean values of algorithms. In this case, the ANOVA test is followed by post hoc tests in order to investigate more accurately. One of the common Post hoc tests is Tukey's test.

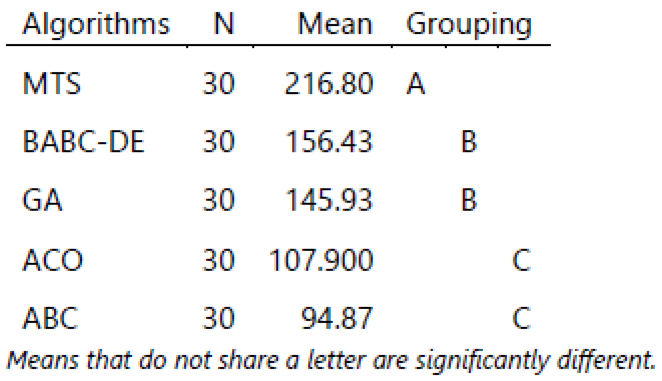
ANOVA test

The numerical example of NMPN was solved 30 times by using the genetic, ACO, ABC, BABC-DE, and MTS algorithms in order to compare the robustness and efficiency of each algorithm. The goal of this experiment is to investigate the performance of the algorithms on this class of problems.
The best value, mean value, worst value, mean CPU time, the standard deviation (SD), and the number times of power reinforcement of t-agent's on node v_e (NRE) in 30 implementations of each algorithm are presented in Table (7).

In order to perform statistical analysis, the ANOVA test was used to evaluate the performance of algorithms. The ANOVA test with the

Table 7: Obtained results for solving the numerical example in 30 implementations by the genetic, ABC, ACO, BABC-DE, MTS algorithms

Algorithm	Best value	Mean value	Worst value	Mean CPU time (seconds)	SD	NRE
GA	212	145.93	104	200.6907	27.5	25
ACO algorithm	116	107.5	107.900	226.4946	2.746	30
ABC algorithm	108	94.87	81	427.2302	8.10	30
BABC-DE algorithm	233	156.43	104	150.500824	41.85	20
MTS algorithm	236	216.80	108	894.927324	29.62	2



Algorithms	N	Mean	Grouping		
MTS	30	216.80	A		
BABC-DE	30	156.43		B	
GA	30	145.93		B	
ACO	30	107.900			C
ABC	30	94.87			C

Means that do not share a letter are significantly different.

Figure 8: Grouping Information Using the Tukey Method

Tukey post hoc test was performed by software Minitab 18 to the simultaneous comparison of the algorithms, and their results are shown in Fig 8. The obtained results of the ANOVA test and Tukey's test are as follows:
The only algorithm in group A is the MTS algorithm and has the highest mean value in 30 iterations of algorithms. The MTS algorithm is significantly better than other algorithms. The GA and the BABC-DE algorithm have not a significant difference from each other. The BABC-DE and genetic algorithms are significantly better than the ACO and ABC algorithms. The ABC algorithm has not a significant difference from the ACO algorithm.

The obtained results from solving the numerical example show that the MTS has better performance than the other algorithms. Here, some of the positive features of the MTS algorithm that may increase the efficiency of the algorithm are pointed out. The MTS algorithm is a local guided search where the chosen moves in recent iterations are rejected in order to avoid trapping in local optimal. This algorithm is designed based on characteristics of NMPN that can find a good approximation of the optimal solution by searching boundaries of $\boldsymbol{FD}$, since the optimal solution of NMPN exists nearby boundary of $\boldsymbol{FD}$. In this paper, a proper evaluation function is introduced to evaluate moves that decrease facilities. Improvement degree of the objective function is used to evaluate moves that increase protecting facilities.

6 Conclusion

In this paper, the NMPN was presented with respect to the possibility of the t-agent's power reinforcement during his onrush (progressing) towards the core. The NMPN is the NP-hard. Thus, the metaheuristic algorithms are a suitable option to solve these problems. In this paper, the genetic, MTS, BABC-DE, ACO, and ABC algorithms were examined in order to find an appropriate algorithm to solve this class of problems. Also, a new method was proposed to obtain heuristic information in the ACO algorithm for solving the NMPN. In the step of decreasing of protecting facilities, a new evaluation function was proposed for Uno's tabu search algorithm. In order to evaluate the performance of the proposed meta-heuristic algorithms, a test problem was randomly generated. The Taguchi method was used for parameters tuning of the mentioned metaheuristic algorithms for solving the test problem. The ANOVA test and Tukey's test were used to compare the mean of obtained solutions of these algorithms in 30 iterations. The ANOVA test rejects the equality assumption of means. Tukey's test shows that the MTS algorithm is significantly better than other algorithms, whereas the ACO and ABC algorithms are significantly worse than other algorithms. The genetic and BABC-DE algorithms are better than the ACO and ABC algorithms but are worse than the MTS algorithm.

References

[1] J. Antony and M. Kaye, *Experimental Quality: A Strategic Approach to Achieve and Improve Quality*, Kluwer Academic, Norwell (1999).

[2] C. O. Astorquiza, I. Contreras and G. Laporte, Multi-level facility location problem, *European Journal of Operational Research*, 267 (2018), 791-805.

[3] O. Berman, T. Drezner, Z. Drezner and G. O. Wesolowsky, A protecting maximal covering problem on a network, *International Transactions in Operational Research*, 16 (2009), 69-86.

[4] J. Cao, B. Yin, Y. Lu, X. Kang and A. Chen, Modified artificial bee colony approach for the 0-1 knapsack problem, *Applied Intelligence*, 48 (2017), 1582-1595.

[5] T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts (1990).

[6] S. Dempe, *Foundations of Bilevel Programming*, Kluwer Academic, Dordrecht (2002).

[7] M. Dorigo, *Optimization, Learning and Natural Algorithms*, Ph.D. thesis, Politecnico di Milano, Italy (1992).

[8] Z. Drezner, Competitive location strategies for two facilities, *Regional Science and Urban Economics*, 12 (1982), 485-493.

[9] B. C. Eaton and R. G. Lipsey, The principle of minimum differentiation reconsidered: Some new developments in the theory of spatial competition, *The Review of Economic Studies*, 42 (1975), 27-49.

[10] F. Ferdowsi, H. R. Maleki and S. Rivaz, Air refueling tanker allocation based on a multi-objective zero-one integer programming model, *Operational Research*, 20 (2018), 1913-1938.

[11] F. Ferdowsi , H. R. Maleki and S. Rivaz, A bi-objective formulation for refueling stations selection problem, *Journal of Mathematical Extension*, 13 (2019), 71-85.

[12] S. L. Hakimi, On locating new facilities in a competitive environment, *European Journal of Operational Research*, 12 (1983), 29-35.

[13] M. A. HashemiNezhad and M. Abbasi, Stochastic capacitated p-median problem with normal distribution, *Journal of Mathematical Extension*, 13 (2019), 1-21.

[14] J. Holland, *Adaptation in Natural and Artificial System*, MIT Press, Cambridge (1992).

[15] H. Hotelling, Stability in competition, *The Economic Journal*, 30 (1929), 41-57.

[16] C. Hu, X. Liu and j. Lu, A bi-objective two-stage robust location model for waste-to-energy facilities under uncertainty, *Decision Support Systems*, 99 (2017), 37-50.

[17] D. Karaboga, *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Technical Report-TR06, Erciyes University, Kayseri (2005).

[18] K. Karkazis, Facilities location in a competitive environment, *European Journal of Operational Research*, 42 (1989), 294-304.

[19] R. Khandouzi, M. R. Peyghami and H. R. Maleki, Solving continuous single-objective defensive location problem based on hybrid directed tabu search algorithm, *International Journal of Advanced Manufacturing Technology*, 76 (2015), 295-310.

[20] D. Kress and E. Pesch, Sequential competitive location on networks, *European Journal of Operational Research*, 217 (2012), 483-499.

[21] H. R. Maleki , R. khanduzi and R. Akbari, A novel hybrid algorithm for solving continuous single objective protecting location problem, *Neural Computing and Applications*, 28 (2016), 3323-3340.

[22] C. S. ReVelle and H. A. Eiselt, Location analysis: A synthesis and survey, *European Journal of Operational Research*, 165 (2005), 1-19.

[23] M. Sakawa, K. Kato and T. Shibano, An interactive Fuzzy satisficing method for multiobjective multidimensional 0-1 knapsack problems through genetic algorithms, *Proceedings of IEEE International Conference on Evolutionary Computation*, (1996), 243-246.

[24] R. Storn and K . Price, Differential evolution- a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, 11(1997), 341-359.

[25] G. Taguchi, *Introduction to Quality Engineering*, Asian Productivity Organization/UNIPUB, White Plains, (1986).

[26] E. G. Talbi, *Metaheuristics: From Design to Implementation*, John Wiley and Sons, Hoboken, (2009).

[27] C. Teye , G. H. Bell and C. J. Bliemer, Entropy maximizing facility location model for port city intermodal terminals, *Transportation Research Part E: Logistics and Transportation Review*, 100 (2017), 1-16.

[28] T. Uno and H. Katagiri, Single and multi-objective protecting location problems on a network, *European Journal of Operational Research*, 188 (2008), 76-84.

[29] T. Uno and K. Kato, An interactive fuzzy satisficing method for multiobjective stochastic protecting location problems, *IEEE International Conference on Fuzzy Systems*, (2011), 879-883.

[30] A. Weber, *Reine Theory des Standorts*, Verlag des Mohr, Tubingen, (1909).

[31] R. Zanjirani and M. Hekmatfar, *Facility Location: Concepts, Models, Algorithms and Case Studies*, Physica-Verlag, Heidelberg, (2009).

[32] R. Zanjirani, S. Fallah, R. Ruise, S. Hosseini and N. Asgari, OR models in urban service facility location: A critical review of applications and future developments, *European Journal of Operational Research*, 726 (2018), 1-27.

Hamid Reza Maleki
Professor of Mathematics
Department of Mathematics
Faculty of Basic Science
Shiraz University of Technology

Shiraz, Iran
E-mail: maleki@sutech.ac.ir

Zahra Maleki
Ph.D. Student of Mathematics
Department of Mathematics
Shiraz University of Technology
Shiraz, Iran
E-mail: z.maleki@sutech.ac.ir

Reza Akbari
Associate Professor of Computer Engineering
Faculty of Computer Engineering and Information Technology
Shiraz University of Technology
Shiraz, Iran
E-mail: akbari@sutech.ac.ir