Iranian Journal of Fuzzy Systems

Volume 19, Number 6, (2022), pp. 51-60

Original paper

University of Sistan and Baluchestan

A new model to protect an important node against two threatening agents

Z. Maleki[1], H. R. Maleki[2] and R. Akbari[3]

[1,2] *Department of Mathematics, Shiraz University of Technology, Shiraz, Iran*

[3] *Department of Computer Engineering and Information Technology, Shiraz University of Technology, Shiraz, Iran*

z.maleki@sutech.ac.ir, maleki@sutech.ac.ir, akbari@sutech.ac.ir

Abstract

One of the main goals of network planners is the protection of important nodes in a network against natural disasters, security threats, attacks, and so on. Given the importance of this issue, a new model is presented in this paper for protecting an important node in a typical network based on a defensive location problem where the two agents threaten this node. The protecting facilities location problem with two agents is formulated as a three-level programming problem. The decision maker in the upper level is a network planner agent. The planner agent wants to find the best possible location of protecting facilities to protect the important node against threatening agents. The second and third levels problems are stated as the shortest path problems in the network in which the edges are weighted with positive values. In this work, the genetic, variable neighborhood search, simulated annealing algorithms are used to solve the problem. The performance of the used metaheuristic algorithms on this class of problems is investigated by a test problem that is generated randomly. Then, t-test are used to compare the performance of these algorithms. The best results are obtained by the variable neighborhood search algorithm.

Keywords: Facilities location, three-level programming problem, meta-heuristic algorithms.

1 Introduction

Protecting the important nodes against threatening agents (t-agents) is one of the important tasks in network planning. Usually, in real cases, huge budgets are usually considered to protect these facilities. A proper locating of protecting facilities may increase the ability to protect these important and sensitive nodes. To have a proper locating of protecting facilities, the threatening agents must be identified, and their behavior must be predicted. In the following, some of the research that has been done in this field are reviewed briefly. In these researches, problems are formulated as a bi-level programming problem. In these problems, two decision-makers exist that decide, respectivly. The first decision-maker is called the defender or leader and the second decision-maker is called invader or follower.

In 2008, Uno and Katagiri [7] introduced the defensive location problem. The defensive location problem (DLP) involves the location of the defensive facilities on vertices of a network to prevent an invader from reaching a strategic site which is called core. Against, the invader selects a proper path to achieve the core with respect to sites of defensive facilities that are determined by the defender. Thus, DLP is formulated as a bi-level programming problem.

In 2015, Khanduzi et al. [2] perused the continuous single-objective defensive location problem (CDLP) and proposed a hybrid algorithm to solve it. In this problem, a decision-maker intends 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 new algorithm that is hybridization of the tabu search algorithm and the Levenberg-Marquardt algorithm.

In 2016, Maleki et al. [4] suggested a novel hybrid algorithm for solving CDLP that integrates the imperialist competitive algorithm and Boyden-Fletcher-Golden-Shano (BFGS) algorithm. The obtained numerical results in this paper have shown that the hybrid algorithm can find high-quality solutions for CDLP problems.

Corresponding Author: H. R. Maleki

Received: October 2021; Revised: June 2022; Accepted: July 2022.

https://doi.org/10.22111/IJFS.2022.7209

In this paper, a protecting facilities location problem with two threatening agents (PFLPTA) based on DLP is stated where the planner agent (p-agent) wants to protect from a target node against two threatening agents (t-agents). In other words, the p-agent intends to keep the t-agents away from the target node as much as possible. For this purpose, the p-agent locates protecting facilities with specified protecting capacity on some vertices sites with respect to existing constraints. Against this, the t-agents want to achieve the target node or closest node to the target node as much as possible. Each t-agent for moving on the edges of the network consumes energy. In addition, the energy of t-agents is reduced in dealing with the protecting facilities of the p-agent. The t-agents move towards the target node until they have energy. Therefore, t-agents choose a path that can approach them to the target node or the closest possible node to the target node. Also, it is assumed that the t-agents move respectively. The t-agent who first moves on the edges of the network destroys the p-agent's protecting facilities along its path towards the target node. So, the second t-agent faces a network on which the p-agent's protecting facilities may be reduced by first t-agent. Hence, this problem is formulated as a three-level programming problem that the first decision-maker is the p-agent and the second and third decision-makers are the first and second t-agents, respectively.

The PFLPTA may be used in various fields, such as computer networks security, flood control in rivers, and so on. In the following, a flood control problem in some rivers is stated as a PFLPTA. Suppose that a city exists on the bank of river C that must be protected from flood, and river C is created from the merging of two rivers A and B, respectively. In addition, suppose that dams A and B were built on rivers A and B, respectively. Thus, there is a possibility the breaking the two dams or opening the emergency gates of two dams at two different times but sufficiently close. Using flood diversion channels is a proper option to control floods and prevent from entering flood into the city. This problem can be interpreted as a PFLPTA, that the p-agent is river engineers and flood in the rivers A and B are t-agent A and t-agent B, respectively. The initial energy amount of the t-agents A and B is equal to the amount of water volume that is entered through the flood into rivers A and B, respectively.

Metaheuristic algorithms are efficient algorithms to generate acceptable solutions for NP-hard problems at a reasonable time. The PFLPTA is NP-hard. Hence, in this paper, metaheuristic algorithms are used to obtain the proper solutions for this problem. In this paper, the genetic algorithm (GA), variable neighborhood search (VNS) algorithm, simulated annealing algorithm (SA) are used for solving the PFLPTA.

The innovations in this paper are as follows: The PFLPTA has been stated and modeled. The PFLPTA is solved by SA, VNS, and genetic algorithms. A new structure is used to generate neighbor solutions in the SA algorithm. To solve the PFLPTA by the VNS algorithm, three new structures for the neighborhood definition have been applied. Also, the Taguchi method is applied to tune parameters of the SA and GA algorithms.

The remaining of this paper is organized as follows: The PFLPTA is formulated as a three-level programming problem in Section 2. In Section 3, solving approaches for the PFLPTA are described in details. In Section 4, the Taguchi method is introduced. A computational experiment is performed in Section 5. Conclusions are given in Section 6.

2 Problem statement

In the PFLPTA, the p-agent wants to locate his protecting facilities on some candidate sites such that prevents the two t-agents from achieving an important site. Here, it is assumed that the p-agent locates his protecting facilities on nodes of a network that the target node is a node of this network, and the t-agents' objective is to achieve the target node or the closest possible node to the target node. In this paper, it is assumed that two t-agents move, respectively. Without loss of generality, the t-agent who moves first is called t-agent A and t-agent who moves after the t-agent A is called t-agent B. The notations and related assumptions to PFLPTA are presented in the following:

Notations and assumptions

Sets

G: The network that the p-agent locates his protecting facilities on it.
$\boldsymbol{V}$: Nodes set of the network G and $\boldsymbol{V} = \{v_1, ...v_n\}$.
$\boldsymbol{E}$: Edges set of the network G and $\mid \boldsymbol{E} \mid = r$.
$\mathbf{p}_{vt}$: Set of all paths from node v to target node.

Decision vectors

• The vector $\boldsymbol{q} = (q_1, q_2, ..., q_{n-2})$ is the decision vector of the p-agent. The value q_i determines whether a protecting facility is placed on node i or not. If there is no the protecting facility on node i, then $q_i = 0$. Otherwise, $q_i = 1$.
• The vector $\boldsymbol{p}_A$ is the decision vector of t-agent A. This vector determines the path that t-agent A selects for reaching

the target node or the closest possible node to the target node t.
• The vector $\boldsymbol{p}_B$ is the decision vector of t-agent B. This vector determines the path that t-agent B selects for reaching target node or the closest possible node to the target node t.

Parameters and Notations

ξ_1: The node of G that t-agent A is on it (the n-th node of G).
ξ_2: The node of G that t-agent B is on it (the (n-1)-th node of G).
t: The node of G that the p-agent protects from it against the t-agents.
e_{ij}: The connector edge between two nodes v_i and v_j.
w_{ij}: Weight of edge e_{ij}, which is a positive value.
$\bar{\alpha}_A$: The initial energy amount of the t-agent A.
$\bar{\alpha}_B$: The initial energy amount of the t-agent B.
$e^l(\boldsymbol{p}_A)$: The l-th edge in the selected path by t-agent A.
$e^l(\boldsymbol{p}_B)$: The l-th edge in the selected path by t-agent B.
$\alpha_A(v^\lambda(\mathbf{p}_A|\mathbf{q}))$: The energy of the t-agent A in the λ-th node of its path toward the target node.
$\alpha_B(v^\lambda(\mathbf{p}_B|\mathbf{q},\mathbf{p}_A))$: The energy of the t-agent B in the λ-th node of its path toward the target node.
β: Capacity of each protecting facility which p-agent locates them on nodes of the network.

Assumptions

• The p-agent can locate his protecting facilities on each node, except nodes ξ_1 and ξ_2.
• In the PFLPTA, the objective of each t-agent is to reach the target node or the closest possible node to the target node.
• In the protection facilities location problem with two t-agents, the p-agent wants to locate his protecting facilities such that prevents the two t-agents from reaching the target node.
• The distance from the node v to the target node c is obtained as follows:

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

• The initial energy of the t-agent A before leaving the node ξ_1 is $\bar{\alpha}_1$.
• The initial energy of the t-agent B before leaving the node ξ_2 is $\bar{\alpha}_2$.
• The energy of the t-agents consume as follows:
1- When the t-agents pass the edge e_{ij}, their energy is reduced by w_{ij}.
2- When the t-agents encounter a protecting facility during moving toward the target node, the energy of the t-agent is reduced by β units.
Thus, passing through the edge e_{ij} reduces the energy of the t-agent A in the amount $\bar{w}_A = w_{ij} + \beta q_j$.
The energy of the t-agent B is reduced by passing through the edge e_{ij} and reaching the node v_j according to the following relation:

$$\bar{w}_B(e_{ij}) = \begin{cases} w_{ij} & j \in \mathbf{p}_A \\ w_{ij} + \beta q_j & j \notin \mathbf{p}_A \end{cases}$$

• Given that, at first t-agent A and then the t-agent B move to reach the target node, therefore t-agent B may encounter network that the number of protecting facilities of the p-agent is reduced on it by t-agent A.
• The t-agent A is on λ-th node of path $\boldsymbol{p}_A$, if his energy on this node that is obtained from the following relation be nonegative. In other words

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

that $e^l(\boldsymbol{p}_A)$ is l-th edge in path $\boldsymbol{p}_A$.
• The t-agent B is on λ-th node of path $\boldsymbol{p}_B$, if its energy on this node that is obtained from the following relation be nonegative. In other words:

$$\alpha_B(v^\lambda(\boldsymbol{p}_B)|\boldsymbol{q},\ \boldsymbol{p}_A) = \bar{\alpha} - \sum_{l=1}^{k} \bar{w}(e^l(\boldsymbol{p}_B)|\boldsymbol{q},\boldsymbol{p}_A) \geq 0, \tag{2}$$

that $e^l(\boldsymbol{p}_B)$ is l-th edge in path $\boldsymbol{p}_B$.
• t-agents vanish if their energy reaches zero or less.

2.1 Mathematical model

In the following this section, the objective functions of p-agent and t-agents are formulated and feasible set of p-agent and t-agents is described. Eventually, PFLPTA is formulated in the final section. Given that the goal of each t-agent is to reach the target node or the closest possible node to the target node, the objective function of t-agents can be formulated as follows:

$$f_A^I(\boldsymbol{q}, \boldsymbol{p}_A) = \min_{v \in \mathbf{p}_A} \{d^c(v) |\ \alpha_A(v|\boldsymbol{q}) \geq 0\},$$
$$f_B^I(\boldsymbol{q}, \boldsymbol{p}_A, \boldsymbol{p}_B) = \min_{v \in \mathbf{p}_B} \{d^c(v) |\ \alpha_B(v|\boldsymbol{q}, \boldsymbol{p}_A) \geq 0\}.$$

Since the p-agent's goal is to keep the t-agents away from the target node, thus the p-agent has two objective functions $f_A^d(\boldsymbol{q}, \boldsymbol{p}_A)$ and $f_B^d(\boldsymbol{q}, \boldsymbol{p}_A, \boldsymbol{p}_B)$ that are obtained by the following relations:

$$\begin{aligned} f_A^d(\boldsymbol{q}, \boldsymbol{p}_A) &= f_A^I(\boldsymbol{q}, \boldsymbol{p}_A), \\ f_B^I(\boldsymbol{q}, \boldsymbol{p}_A, \boldsymbol{p}_B) &= f_B^d(\boldsymbol{q}, \boldsymbol{p}_A, \boldsymbol{p}_B). \end{aligned} \tag{3}$$

There are many constraints in facilities location. For example, the cost of facilities building and allocation of staff can be constraints of p-agent in facilities location. In this paper, constraints of PFLPTA are considered as linear. Assume that m denotes the number of the constraints of the PFLPTA, the matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n-2}$ and vector $\boldsymbol{b} \in \mathbb{R}^m$ be coefficients matrix and right-hand side vector, respectively. Thus, the feasible set of the p-agent is represented as follows:

$$\boldsymbol{SD} = \{\boldsymbol{q} = (q_1, ..., q_{n-2}) |\ \boldsymbol{A}\boldsymbol{q} \leq \boldsymbol{b},\ q_i \in \{0, 1\}\}.$$

Feasible solutions set of t-agent A is represented by $\boldsymbol{SI}_A$ and is consisted of all paths from node ξ_1 to target node t. Also, feasible solutions set of t-agent B is represented by $\boldsymbol{SI}_B$ and is consisted of all paths from node ξ_2 to target node t. PFLPTA is formulated as follows:

$$\begin{aligned} &\max_{\boldsymbol{q}}\ f^d(\boldsymbol{q}, \boldsymbol{p}_A) \\ &\max_{\boldsymbol{q}}\ f^d(\boldsymbol{q}, \boldsymbol{p}_A, \boldsymbol{p}_B) \\ &\text{where } \boldsymbol{p}_A \text{ and } \boldsymbol{p}_B \text{ solve} \\ &\min_{\boldsymbol{p}_A}\ f^d(\boldsymbol{q}, \boldsymbol{p}_A) \\ &\text{where } \boldsymbol{p}_B \text{ solves} \\ &\min_{\boldsymbol{p}_B}\ f^d(\boldsymbol{q}, \boldsymbol{p}_A, \boldsymbol{p}_B) \\ &\text{s.t.} \quad \boldsymbol{q} \in \boldsymbol{SD},\ \boldsymbol{p}_A \in \boldsymbol{SI}_A \text{ and } \boldsymbol{p}_B \in \boldsymbol{SI}_B \end{aligned} \tag{4}$$

3 Proposed methology for solving PFLPTA

A method to solve the multi-objective programming problems is the fuzzy programming approach. The first level of problem 4 is a two objective problem that is converted to following optimization problem by fuzzy programming approach:

$$\begin{aligned} &\max \quad \lambda, \\ &\text{s.t.} \quad \mu_A(f_A^d(\boldsymbol{q}, \boldsymbol{p}_A)) \geq \lambda, \\ &\qquad \mu_B(f_B^d(\boldsymbol{q}, \boldsymbol{p}_A, \boldsymbol{p}_B)) \geq \lambda, \end{aligned} \tag{5}$$

where μ_A and μ_B are membership functions of objective functions $f_A^d(\boldsymbol{q}, \boldsymbol{p}_A)$ and $f_B^d(\boldsymbol{q}, \boldsymbol{p}_A, \boldsymbol{p}_B)$ that are defined as follows:

$$\mu_A(f_A^d(\mathbf{q}, \mathbf{p}_A)) = \frac{f_A^d(\mathbf{q}, \mathbf{p}_A)}{d^c(\xi_1)},$$
$$\mu_B(f_B^d(\mathbf{q}, \mathbf{p}_A, \mathbf{p}_B)) = \frac{f_B^d(\mathbf{q}, \mathbf{p}_A, \mathbf{p}_B)}{d^c(\xi_2)}.$$

Therefore, solving Problem 4 is equivalent to solve the following problem:

$$\begin{aligned}
&\max\ \lambda \\
&\quad \mu_A(f_A^d(\boldsymbol{q},\boldsymbol{p}_A)) \geq \lambda \\
&\quad\quad \mu_B(f_B^d(\boldsymbol{q},\boldsymbol{p}_A,\boldsymbol{p}_B)) \geq \lambda \\
&\text{where } \boldsymbol{p}_A \text{ and } \boldsymbol{p}_B \text{ solve} \\
&\min_{\boldsymbol{p}_A}\ f^d(\boldsymbol{q},\boldsymbol{p}_A) \\
&\text{where } \boldsymbol{p}_B \text{ solves} \\
&\min_{\boldsymbol{p}_B}\ f^d(\boldsymbol{q},\boldsymbol{p}_A,\boldsymbol{p}_B) \\
&\text{s.t.}\quad \boldsymbol{q} \in \boldsymbol{SD},\ \boldsymbol{p}_A \in \boldsymbol{SI}_A \text{ and } \boldsymbol{p}_B \in \boldsymbol{SI}_B
\end{aligned} \tag{6}$$

In this paper, a hybrid method is proposed to solve the Problem 6. The second and third levels problems are solved by exact algorithms and first level is solved by a meta-heuristic algorithm, in this method. To obtain the values of $f_A^d(\boldsymbol{q},\boldsymbol{p}_A)$ and $f_B^d(\boldsymbol{q},\boldsymbol{p}_A,\boldsymbol{p}_B)$, we need to find following values:

- The minimum weighted sums from ξ_1 to all the vertices for location $\mathbf{q}$.
- The minimum weighted sums from ξ_2 to all the vertices given that vector components $\mathbf{q}$ were updated as follows:

$$q_j = \begin{cases} 0 & j \in \boldsymbol{p}_A \text{ and } \alpha_{\mathrm{A}}(\mathrm{v_j}|\ \boldsymbol{q},\boldsymbol{p}_{\mathrm{A}}) \geq 0 \\ q_j & j \notin \boldsymbol{p}_A \end{cases} \tag{7}$$

- The sum of the minimum weights from all vertices to target node t with this assumption that any protecting facility is not located on network vertices.

Using the shortest path algorithms is an appropriate option to find each of the above values. Due to the positive weight of the edges in the network, Dijkstra algorithm is a suitable algorithm for finding the shortest path from a node to another node in the network.

Since the PFLPTA is an extension of the defensive location problem and the defensive location problem is NP-hard [7], thus the PFLPTA is NP-hard. Hence, exact methods are not efficient for solving a large-scale PFLPTA. In this situation, meta-heuristic algorithms are a proper alternative for solving PFLPTA. In this paper, the genetic, SA and VNS algorithms are used for solving the PFLPTA.

3.1 Genetic algorithm

This algorithm is an optimization algorithm that is developed by Holland [1]. The Genetic algorithm (GA) starts with a random initial population that makes evolve by applying a set of stochastic operators on some individual of population. These stochastic operators are selection, mutation and crossover. The GA proceeds by generating successive generations of better individuals that are generated by mentioned stochastic operators. In the following, the GA is described for solving PFLPTA.

GA for solving PFLPTA

In this section, solving of PFLPTA is investigated by GA. The process of fitting the problem to the GA is begun by defining a chromosome (solution) as an array of variables with binary values. A chromosome in the GA for solving PFLPTA with variables $q_1, ..., q_{n-2}$ can be shown as $(q_1, ...q_{n-2})$. For solving PFLPTA by GA, the fitness of each solution $\boldsymbol{q} = (q_1, ..., q_{n-2})$ is described by $\min(\mu_A(f_A^d(\boldsymbol{q}, p_A)), \mu_B(f_B^d(\boldsymbol{q}, p_A, p_B)))$. In the following, the required components for designing the GA that are used to solve PFLPTA are explained briefly. The GA begins with a population of solutions that must be evaluated by mentioned fitness function. In order to generate a population with higher fitness, some of individuals are selected by selection operator (In this study, we use roulette wheel selection operator). Afterwards, offspring are generated by crossover and mutation operators. The implementation of the GA to solve the PFLPTA is as follows:

3.2 SA algorithm

This algorithm is a an algorithmic means that is developed by Kirkpatrick in 1983 [3].The SA algorithm uses from annealing principle to search global optimum solution for discrete optimization problems. In the annealing process,

Input: Crossover rate (p_c), mutation rate (p_m), size of the population (N_{pop}), number of the objective function evaluations (NEF) and input parameters of the PFLPTA.
Initialize a population of solutions randomly.
Evaluate each member of population.
while *stopping criterion is not satisfied* **do**
Select parents;
Recombine pairs of parents by crossover operator;
Mutate some offsprings by mutation operator;
Evaluate new candidate individuals;
Select individuals for next generation.
end

Algorithm 1: Pseudo code for the GA

crystalline solid is heated in order to increase the internal energy. Afterward, the cooling process is gradually performed to reach its most regular crystal lattice configuration possible that is free of crystal defects. In this paper, SA algorithm is used to solve PFLPTA that is explined summarily in the following.

SA algorithm for solving PFLPTA

The SA algorithm begins with generating a random initial solution. In this study, a neighbor solution is generated by changing an element of solution string from 1 to 0 and an element from 0 to 1. If the current solution is equal to zero vector, an element of this vector is changed its value to 1. These elements are selected randomly. The algorithm moves to the neighbor solution provided that its fitness is equal or better than the current solution. Otherwise, moving to neighbor solution is done with a probability. The probability of accepting non-improving solutions is typically non-increasing with each iteration of the algorithm and depends on the temperature. Here, this probability is calculated as follows:

$$p(\Delta f, T) = \exp(\frac{-\Delta f}{T}) = \exp(-\frac{f(q^{'}) - f(q)}{T}),$$

where q' and q are neighbor and current solutions, respectively. Also, parameter T is temperature in current iteration. The mentioned process is repeated in a specified number of iterations. Then, the temperature is decreased according to the cooling schedule. One of the most common cooling schedules is geometric cooling schedule in which the temperature is updated as $T = \alpha \times T$ where $0 < \alpha < 1$. This process is repeated in the new temperature until reaching a stopping condition. In this paper, the stopping criterion is the number of objective function evaluations. The SA algorithm is briefed in Fig. 2.

Input: Number of iterations which the temperature remains constant (SubIt), initial temperature (T_0), cooling rate (α) .
$\boldsymbol{q}$=Random initial solution.
Let $T = T_0 \geq 0$.
while *stopping criterion is not satisfied* **do**
for $i = 0$ *to subitIt* **do**
Generate a neighbor solution $\boldsymbol{q}'$ from solution $\boldsymbol{q}$.
Calculate $\triangle_{\boldsymbol{q},\boldsymbol{q}'} = f(\boldsymbol{q}') - f(\boldsymbol{q})$.
if $\triangle_{\boldsymbol{q},\boldsymbol{q}'} \geq 0$ **then**
$\boldsymbol{q} \longleftarrow \boldsymbol{q}'$.
else
$\boldsymbol{q} \longleftarrow \boldsymbol{q}'$ with probability $\exp(-\triangle_{\boldsymbol{q},\boldsymbol{q}'})/T$.
end
end
$T = \alpha T$
end

Algorithm 2: Pseudo code for the SA algorithm

3.3 VNS algorithm

The VNS is another single-solution meta-heuristic algorithm which is proposed by Mladenovic [5]. Exploring neighbor solutions that are generated by different neighborhood structures is a basic feature of the VNS algorithm. Thus, we

propose a set of neighbor search operators for this algorithm. In the following of this section, the VNS algorithm is investigated for solving PFLPTA.

VNS algorithm for solving PFLPTA

In this paper, to solve the PFLPTA using the VNS algorithm, three neighborhood structures are considered, which are explained in the following:

Neighborhood structure N_1: In this neighborhood structure, the neighbor solution $\boldsymbol{q}'$ in the neighborhood $N_1(\boldsymbol{q})$ is obtained by applying the flip operator on an element of solution $\boldsymbol{q}$ that this elament is selected randomly.

Neighborhood structure N_2: To generate the neighbor solution $\boldsymbol{q}'$ in the neighborhood $N_2(\boldsymbol{q})$, two elements of the solution $\boldsymbol{q}$ with values 0 and 1 are selected randomly and then these values are converted to 1 and 0, respectively. Also, If q is zero vector, a neighbor solution is obtained by applying a flip operator on an element of q.

Neighborhood structure N_3: Solution $\boldsymbol{q}'$ in neighborhood $N_3(\boldsymbol{q})$ is obtained by selecting two elements from solution $\boldsymbol{q}$ randomly and inverting the sequence of elements between these two elements.

Assume that, a solution $\boldsymbol{q}'$ is generated in the current neighborhood $N_k(\boldsymbol{q})$ randomly. Then, a local search procedure is applied in order to generate the solution $\boldsymbol{q}"$. If the fitness of solution $\boldsymbol{q}"$ is better than fitness of solution $\boldsymbol{q}$, then solution $\boldsymbol{q}$ is replaced by the solution $\boldsymbol{q}''$ and the same search procedure is restarted with neighborhood $N_1(\boldsymbol{q})$. Otherwise, in order to generate a new solution and attempt to improve it, algorithm moves to the next neighborhood structure i.e. $N_{k+1}(\boldsymbol{q})$.

Here, neighborhood N_1 is used in the local search process and stopping criterion is the number of objective function evaluations. Fig. 3 represents Pseudo code the of the VNS algorithm.

Input: A set of neighborhood structures N_k for $k \in \{1, ..., k_{\max}\}$ and number of iterations in local search process (SubIt)
Generate the initial solution $\boldsymbol{q}$.
Let $\boldsymbol{q} = \boldsymbol{q}_0$.
repeat
  $k = 1$.
  repeat
    Select solution $\boldsymbol{q}'$ from set $N_k(\boldsymbol{q})$ randomly.
    $\boldsymbol{q}"$=local search ($\boldsymbol{q}'$).
    if $f(\boldsymbol{q}") \leq f(\boldsymbol{q})$ **then**
      $\boldsymbol{q} = \boldsymbol{q}"$.
      $k = 1$.
    else
      $k = k + 1$
    end
  until $k < k_{max}$;
until *stopping criterion is satisfied* ;

Algorithm 3: Pseudo code for the VNS algorithm

4 Computational study

The purpose of a computational experiment is for the validation of the model and investigation of the algorithms behavior to solve this class of problems . This example is solved by three metaheuristic algorithms which are the genetic, VNS and SA 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.

4.1 Test problem

Consider a network with $n = 200$ nodes and 985 edges that weight of edges is randomly selected from integer numbers in interval $[1, 30]$. The initial energy amount of the first t-agent and the second t-agent is considered $\bar{\alpha}_{\mathrm{A}} = 200$ and $\bar{\alpha}_{\mathrm{B}} = 210$, respectively. Also, it is assumed that the capacity of each protecting facility is 50. For all $i = 1, ..., m$ and $j = 1, ..., n-2$, the elements a_{ij} of coefficients matrix $\boldsymbol{A}^{m\times n-2}$ are scalars that are randomly chosen from set $\{1, .., 30\}$. The elements of the right-hand side vector $\boldsymbol{b}$ are determined as follows:

$$b_i = a + rand(0, 1)(b - a)\sum_{j=1}^{n-2} a_{ij}, \quad i = 1, ..., m,$$

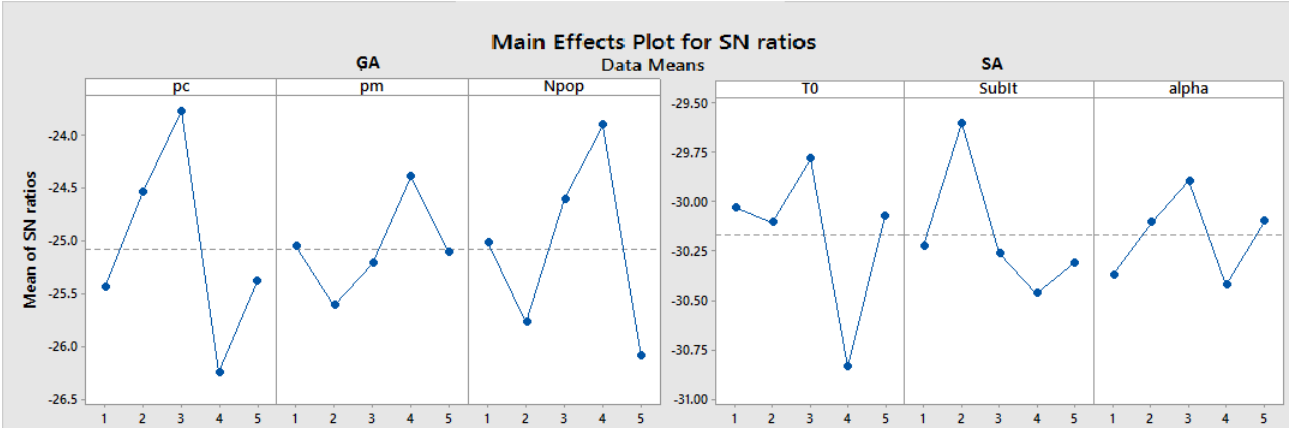

Figure 1: Parameter setting of GA and SA

where $rand(0,1)$ is a random number in interval (0,1) and values of variables a and b are 0.1 and 0.2, respectively. For comparing the algorithms fairly, the stopping condition for each algorithm is considered the number of objective function evaluations (NFE). For all of the used metaheuristic algorithms in this paper, we consider NFE=1000.

4.2 Parameter setting

The performance of meta-heuristic algorithms depend on the values of parameters. One of the most common methods to tune parameters a metaheuristic algorithm is Taguchi method. For a good introduction of Taguchi method, we refer the readers to [6]. The SA, VNS and genetic algorithms have 3, 1 and 3 controllable parameters, respectively. The SA and GA algorithms with each combination of parameters that is suggested by Taguchi method was run three times to obtain more reliable results. The VNS algorithm has a controllable parameter. Thus, we determined the value of parameter $SubIt$ with full factorial design. To obtain more reliable results, the VNS algorithm was run three times with each suggested level of parameter $SubIt$. According to the average of the values of the objective function in each experiment, the best value for the parameter $SubIt$ was selected.

The tested values and suggested values for controllable parameters of the SA, VNS and genetic algorithms are shown in Table 1. In addition to, mean S/N ratio plot for each level of the parameters in genetic and SA algorithms are shown in Figure 1.

Table 1: Selected and suggested values for parameters of VNS, SA and genetic algorithms

Algorithm	Parameter	Tested values	Suggested value
SA	T_0	50,150,250,350,450	250
	SubIt	10,20,30,40,50	20
	α	0.8,0.8475,0.895,0.9425,0.99	0.895
GA	p_c	0.65,0.7,0.75,0.8,0.85	0.75
	p_m	0.15,0.2,0.25,0.3,0.35	0.3
	N_{pop}	10,30,50,70,90	70
VNS	SubIt	10,20,30,40,50	40

4.3 Evaluation of results

In this paper, a statistical analysis is used to compare performance of meta-heuristic optimization algorithms that are used to solve test problem. The t-test is a statistical analysis that can use to specify whether the difference between mean values of two meta-heuristic algorithms is statistically significant or not.

4.3.1 t- test

The t-test can be used in hypothesis testing that compares the performance of two algorithms. Here, the null hypothesis and alternative hypothesis considered to compare two meta-heuristic algorithms by t-test is as follows:

$$\begin{cases} H_0 : \ \mu_1 = \mu_2 \\ H_1 : \ \mu_1 > \mu_2 \end{cases} \tag{8}$$

where μ_1 and μ_2 are results mean of algorithms 1 and 2, respectively. In t-test, the difference between meta-heuristic algorithms are determined by P-Value. The performance of algorithm 1 is signicantly better than performance of algorithm 2 in case that P-Value be small (< 0.05).

In order to compare the robustness and efficiency algorithms, VNS, SA and genetic algorithms have run 30 times to solve generated numerical example. The obtained results which include best value, mean value, worst value and mean CPU time have been represented in table 2. In addition, the distance of t-agent A (DTA) and distance t-agent B (DTB)

Table 2: Obtained results for solving the numerical example in 30 implementations of SA, VNS and genetic algorithms

Algorithm	SA	VNS	genetic
Best value	0.3881	0.6117	0.5261
Mean value	0.36056	0.5368	0.46953
Worst value	0.3246	0.4249	0.4142
Mean CPU time	83.3211	74.3466	78.2398

from the target node where the objective function value is best between 30 iterations of algorithm is presented in Table 3. The t-test for each of two algorithms is done by software minitab 18 and the obtained results used to interperet

Table 3: Distance t-agent A and t-agent B of target node where best value of in 30 iteration of algorithms is obtained

Algorithm	SA	VNS	GA
(DTA,DTB)	(108,104)	(167,173)	(158,141)

t-test are shown in fig 2.

Table 4: Obtained P-Value for each paired of algorithms by t-test

Algorithms	VNS and GA	VNS and GA	GA and SA
P-Value	0.000	0.000	0.000
Significance level	0.05	0.05	0.05

In table 4, the first row is the result of comparsion between VNS and GA: Because P-Value is less than 0.05, the VNS is significantly better than GA.
In table 4, the second row is the result of comparsion between the VNS and SA: Because P-Value is less than 0.05, the VNS is significantly better than SA.
In table 4, the third row is the result of comparsion between the GA and SA: Because P-Value is less than 0.05, the GA is significantly better than SA.
Thus, VNS algorithm has the highest mean value between 30 iterations of the algorithm and is significantly better than other algorithms. Also, we can conclude that the GA is significantly better than the SA algorithm and the SA algorithm has the worst performance among the used algorithms. Some features of the VNS algorithm that may increase the efficiency of this algorithm are as follows:
The VNS algorithm is a simple scheme and needs a few parameters. The VNS algorithm explores the various neighborhood structures of the current solution and jumps from there to a new one if and only if an improvement was made. In other words, the VNS algorithm is designed based on this fact that various neighborhoods in local search may generate different local optimal and allow the algorithm to escape the local optimal.

5 Conclusion

In this paper, the PFLPTA was stated and formulated. In this problem p-agent wants to locate protecting facilities to prevent t-agent from reaching the target node. The meta-heuristic algorithms are used for solving this problem. In this

T-Value	DF	P-Value
3.53	57	0.000

VNS and GA

T-Value	DF	P-Value
18.56	58	0.000

VNS and SA

T-Value	DF	P-Value
-12.02	58	0.000

GA and SA

Figure 2: The obtained results of the t-test to compare algorithms VNS and GA, VNS and SA, and GA and SA

paper, the GA, SA, VNS algorithms are applied to solve this problem. In the SA and VNS algorithms, we defined new neighborhood search operators. The Taguchi method was used to tune the parameters of the GA and VNS algorithms. To evaluate the performance of the mentioned metaheuristic algorithms, a numerical example was generated randomly and the performance of these algorithms on this class of problems was compared by t-test. The t-test was shown that the VNS algorithm has better performance than other algorithms.

References

[1] J. Holland, *Adaptation in natural and artifcial system*, MIT Press, Cambridge, 1992.

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

[3] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, *Optimization by simulated annealing*, Science, **220**(4598) (1983), 671-680.

[4] H. R. Maleki, R. Khanduzi, R. Akbari, *A novel hybrid algorithm for solving continuous single objective protecting location problem*, Neural Computing and Applications, **28**(11) (2016), 3323-3340.

[5] N. Mladenovic, P. Hansen, *Variable neighborhood search*, Computers and Operations Research, **24**(11) (1997), 1097-1100.

[6] M. Salehi, H. R. Maleki, S. Niroomand, *Solving a new cost-oriented assembly line balancing problem by classical and hybrid meta-heuristic algorithms*, Neural Computing and Applications, **32** (2020), 8217-8243.

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