A Modified Emperor Penguin Optimizer Algorithm for Solving Fixed-Charged Transshipment Problem

Mohamed Meselhy Eltoukhy[1*], Mohammad Zakaraia[2]
[1]Department of Information Technology, College of Computing and Information Technology at Khulais, University of Jeddah, Jeddah, Saudi Arabia
[2]Faculty of graduate studies for statistical research, Cairo University
E-mail: mmeltoukhy@uj.edu.sa[1], dr.mzakaraia@gmail.com[2]
[*]Corresponding author

Keywords: Fixed-Charged transportation problem, transshipment problem, metaheuristics, Emperor penguin optimizer algorithm, Taguchi's orthogonal arrays, design of experiments, Wilcoxon test, Friedman test

Received: February 8, 2024

This paper introduces a novel optimization problem termed the Fixed Charged Transshipment Problem (FCTP), which incorporates fixed charges for selected routes. A new formulation for this problem is presented, aiming to address the combinatorial nature of the challenge. The study further introduces a Modified Emperor Penguin Optimizer (EPO) algorithm designed to enhance the solution approach. To evaluate the performance of the Modified EPO, a comparative analysis is conducted against the classical EPO and Particle Swarm Optimization (PSO) algorithms. 19 problems, including various multi-modal test optimization functions, serve as the testing ground. Results demonstrate the efficacy of the Modified EPO, establishing its superiority over the classical EPO and PSO. Additionally, a heuristic procedure is proposed for solving the combinatorial aspect of the FCTP. This heuristic is hybridized with both the Modified EPO and PSO algorithms. 30 FCTP problems are generated using a code available at https://github.com/MZakaraia/EPO_Transshipment/. Taguchi's orthogonal arrays are employed to optimize parameter levels for both algorithms. The study concludes with the comparison of the Modified Hybrid EPO and Hybrid PSO in solving the 30 generated FCTP problems. Remarkably, the Modified Hybrid EPO algorithm outperforms the Hybrid PSO, showing its effectiveness in addressing the Fixed Charged Transshipment Problem in terms of means and robustness.

Povzetek: Članek predstavlja spremenjeni algoritem cesarskega pingvina za reševanje problema s fiksnimi stroški prenosa, ki kaže premoč nad klasičnimi metodami z robustnostjo in učinkovitostjo rešitev.

1 Introduction

The fixed charged transshipment problem is a well-known optimization problem in the field of logistics and supply chain management. It involves determining the optimal flow of goods through a network, considering fixed costs associated with transshipments between various nodes. Solving the FCTP efficiently is crucial for optimizing supply chain operations, reducing costs, and enhancing overall system performance. In recent years, nature-inspired optimization algorithms have gained popularity as effective tools for solving complex optimization problems. One such algorithm is the Emperor Penguin Optimizer (EPO), which is inspired by the behavior and social interactions of emperor penguins in their natural habitat. The EPO algorithm is known for its ability to effectively handle continuous and discrete optimization problems. This paper presents a modified version of the Emperor Penguin Optimizer algorithm tailored specifically for addressing the Fixed-Charged Transshipment Problem. The main objective of this study is to investigate the effectiveness and efficiency of the modified EPO algorithm in finding high-quality solutions for the FCTP. So, the contribution of this can be summarized as follows:

- Proposing a new formulation for the fixed charged transshipment problem by considering fixed costs for the routes.
- Adapting a new modification of the population-based metaheuristic EPO algorithm for solving FTCP.
- Generating a dataset of 30 problems for FTCP to validate the proposed EPO algorithm for solving FTCP.

The remainder of this paper is organized as follows: Section 2 provides a literature review of related studies on transshipment problems. Section 3 presents the mathematical formulation of the Fixed Charge Transshipment Problem with a discussion that shows the novelty of the proposed formulation. Section 4 describes the EPO algorithm in detail, while Section 5 introduces the modified EPO and presents comparative results. Section 6 outlines the adapted EPO for solving FCTP. In this Section the computational complexity of the proposed EPO is presented. The experimental design is implemented to optimize the parameters of the EPO and adopted particle swarm optimization algorithm for solving the problem. The computational results for 30 generated problems are performed to compare between the hybrid

EPO and PSO for solving the FCTP. Finally, Section 7 concludes the paper with a summary of the findings, highlighting the advantages and potential applications of the proposed algorithm for solving the Fixed Charge Transshipment Problem.

2 Literature review

This section provides a literature review of prior research on the transshipment problem, offering a comprehensive overview of existing work and highlighting the motivation behind the current study. Herer and Tzur [1] investigated the strategy of transshipments in a dynamic, deterministic demand environment over a finite planning horizon. Their study considered a system of two locations replenished by a single supplier, incorporating various costs and deriving structural properties of optimal policies, leading to the development of an efficient polynomial time algorithm for obtaining the optimal strategy and motivating the adoption of transshipments in replenishment strategies. Reyes [2] used the Shapley value concept from cooperative game theory to solve the transshipment problem and demonstrated its efficacy through a numerical example. Herer et al. [3] examined a supply chain with multiple retailers and a supplier, where they established optimal replenishment and transshipment policies to minimize long-run average costs. Through a sample-path-based optimization procedure, they calculated order-up-to quantities using a linear programming/network flow framework.

Belgasmi et al. [4] examined a multi-location inventory system with centrally coordinated inventory choices, allowing lateral transshipments within the same echelon to reduce costs and improve service level. They proposed a multi-objective model to optimize cost, fill rate, and transshipment lead times. They utilized an evolutionary multi-objective optimization approach to approximate the optimal trade-offs between these conflicting objectives. Sharma and Jana [5] developed a transshipment planning model for the petroleum refinery industry, aiming to minimize costs, maximize production, and meet storage and demand requirements. They employed a fuzzy goal programming (FGP) model with integrated genetic algorithms (GA) to handle imprecision and provide flexible solutions. A case example showcased the effectiveness of this integrated technique in optimizing transshipment operations. Khurana and Arora [6] extended the standard transshipment model to include inequality constraints. Their algorithm transformed the problem into an equivalent transportation problem to obtain the optimal solution. They discussed balanced and unbalanced transshipment problems, emphasizing the algorithm's applicability in addressing distribution problems with mixed constraints and paradoxical situations. Özdemir et al. [7] investigated the coordination of stocking locations considering lateral transshipments and supply capacity in the transshipment model. They formulated the capacitated supply scenario as a network flow problem within a stochastic optimization framework. They found that system behavior depends on production capacity and highlighted the importance of capacity flexibility or transshipment flexibility for maintaining desired service levels in a production-inventory system.

Khurana, et al. [8] developed an algorithm to solve a transshipment problem with the objective of minimizing transportation duration. They transformed the problem into an equivalent transportation problem and obtained the optimal solution. Their algorithm is easy to understand and apply, making it suitable for addressing various products distribution problems. In addition, balanced and unbalanced time minimization scenarios were discussed with numerical examples. Kumar, et al. [9] addressed the challenges of uncertainty in transshipment problems by representing parameters as intuitionistic fuzzy numbers. Their proposed method is based on ambiguity and vagueness indices to derive a fuzzy optimal solution without the need for an initial basic feasible solution. The technique demonstrated computational efficiency and applicability to a wide range of transshipment problems, supported by numerical illustrations. Garg et al. [10] investigated a fuzzy fractional two-stage transshipment problem, using the ratio of costs divided by benefits as the objective function. They employed the extension principle and Charnes-Cooper transformation method to find the fuzzy objective value. The proposed formulation and solution method demonstrated superior efficiency compared to the existing literature.

Table 1 shows the literature review summary. To the best of our knowledge, the fixed-charge transshipment problem, which extends the fixed-charge transportation problem by including transshipment nodes, has not yet been investigated. Therefore, this paper presents a new model for the transshipment problem that incorporates both fixed costs and transportation costs.

Table 1: Literature review summary

Author	Problem	Approach
Herer and Tzur [1]	Dynamic transshipment problem	Heuristic approach
Reyes [2]	Classical transshipment problem	Game theory approach
Herer et al. [3]	Multi-location transshipment problem	Linear programming and network follow
Belgasmi et al. [4]	Multi-objective multi-location transshipment problem	Strength Pareto evolutionary algorithm
Sharma and Jana [5]	Transshipment management problem	Fuzzy goal programming and genetic algorithm
Khurana and Arora [6]	Unbalanced transshipment problem with mixed constraints	Liner Programming
Özdemir et al. [7]	Multi-location transshipment problem with	Random search, Simulation applications, and

	capacitated production	sample average approximate
Khurana, et al. [8]	Time minimizing transshipment problem	Heuristic approach
Kumar, et al. [9]	Fuzzy transshipment problem	Heuristic approach
Garg et al. [10]	Fractional two-stage transshipment problem under uncertainty	Charnes–Cooper transformation method, linear programming

Given the combinatorial complexity of the problem, the classical approaches listed in Table 1, primarily those mentioned, are not well-suited to solving the fixed-charge transshipment problem. Consequently, this paper adopts one of the latest population-based metaheuristics, namely the Emperor Penguin Optimizer Algorithm, to address this challenge.

3 Mathematical formulation

Fixed Charge Transshipment Problem is a combinatorial optimization problem in which a set of products or goods is transported from source nodes to destination nodes through a network of intermediate transshipment nodes. In this problem, there is a fixed cost associated with using each transshipment node, and a variable cost for transporting each unit of product between the nodes. The FCTP can be mathematically modeled as follows:

Notations:

i	Set of sources
j	Set of destinations
k	Set of transshipment nodes
f_k	Fixed cost associated with transshipment node k
c_{ij}	Unit cost of transporting a product from source i to destination j via transshipment node k
x_{ij}	Amount of product transported from source i to destination j directly (without transshipment)
s_i	The available quantity produced by source i
d_j	The required demand by destination j
y_{ijk}	Amount of product transshipped at node k from source i to destination j
z_k	A binary variable representing whether transshipment node k is used or not

Mathematical model:

$$\min \sum_{k\in K} f_k z_k + \sum_{i\in I}\sum_{j\in J} c_{ij}\left(x_{ij} + \sum_{k\in K} y_{ijk}\right) \quad (1)$$

Subject to:

$$\sum_{j\in J}\left(x_{ij} + \sum_{k\in K} y_{ijk}\right) = s_i, \forall i \in I \quad (2)$$

$$\sum_{i\in I}\left(x_{ij} + \sum_{k\in K} y_{ijk}\right) = d_j, \forall j \in J \quad (3)$$

$$\sum_{i\in I}\sum_{j\in J} y_{ijk} \le Q_k z_k, \forall k \in K \quad (4)$$

$$x_{ij} \ge 0, \forall i \in I, j \in J \quad (5)$$

$$y_{ijk} \ge 0, \forall i \in I, j \in J, k \in K \quad (6)$$

$$z_k \in \{0,1\}, \forall k \in K \quad (7)$$

The fixed-charged transshipment problem, as shown in the mathematical model, is a combinatorial optimization problem known for its NP-hardness, making it very difficult to solve using classical approaches. Therefore, metaheuristics are the preferred choice for tackling such problems. These approaches can be broadly classified into two categories: single-based metaheuristics (e.g., simulated annealing, Tabu search, and variable neighborhood algorithms) and population-based metaheuristics (e.g., particle swarm optimization, gray wolf, genetic algorithm, and the proposed emperor penguin optimizer algorithm (EPO) presented in this paper) [11]–[14].

In this paper a new form of the transshipment problem is presented, which considers fixed charges associated with selected routes. To fill this gap, we propose a hybridized EPO algorithm that incorporates a heuristic procedure based on priority vectors to achieve solution improvement. Before presenting the hybridized algorithm, we propose a modified EPO algorithm and compare its performance with both the classical EPO and a particle swarm optimization algorithm.

4 Emperor penguin optimizer algorithm

The emperor penguin optimizer algorithm is a population-based metaheuristic that was first proposed by Dhiman and Kumar [15]. It mimics the emperor penguins' huddling behavior. The steps of the algorithm include generating the boundaries of the huddle, computing the temperature of the huddle, the distances between specific penguins, and finding the emperor penguins by obtaining the effective mover. The proposed algorithm by Dhiman and Kumar [15] consists of four phases:

- Generating positions based on huddle boundaries.
- Calculating the temperature around the huddle.
- Determining the distances between emperor penguins.
- Relocating procedure.

The next subsections show these phases followed by the full pseudo code of the basic POA algorithm.

4.1 Generating positions based on huddle boundaries

In this phase, the positions of the penguins are to be generated using the huddle boundaries, where they are restricted by the lower bound (LB) and the upper bound (UB). So, each penguin position is to be generated using equation (8) for all n penguins in the huddle.

$$Pos_i = LB + rand(0,1)(UB - LB), \forall i = \{1, \dots, n\} \tag{8}$$

4.2 Calculating temperature profile around the huddle

In this phase, the temperature profile of the huddle is to be calculated using the radius of the huddle (R). If the radius of the huddle is less than 1, then the temperature (T) equals to 1, and T equals 0 if R greater than or equals 1. The radius of the huddle in the algorithm is to be generated randomly in each iteration from the interval [0,1]. So, the new temperature profile around the huddle (T') can be calculated according to equation (9).

$$T' = T - \frac{MaxItr}{Itr - MaxItr} \qquad T = \begin{cases} 1, R < 0 \\ 0, R \le 1 \end{cases} \tag{9}$$

4.3 Determining the distance between the emperor penguins

The distance between emperor penguins and the best penguin can be calculated using two vectors that prevent collision $\vec{A}$ and $\vec{C}$, the position of the penguin i in the current iteration ($P_{Itr}(i)$), a social force S, and the position of the current optimal emperor penguin (P_{opt}). The proposed equation by Dhiman and Kumar [15] to calculate the distance (D) is presented in equation (10). The calculations of the two collision vectors $\vec{A}$ and $\vec{C}$ are shown in equations (11) and (12), respectively. The P_{grid} variable found in equation (13) is the absolute value of the difference between the position of the best emperor penguin and the current penguin i, where the equation of the P_{grid} is equation (13). The social force function can be calculated using equation (14).

$$D = \left|S(\vec{A})P_{opt} - \vec{C}P_{Itr}(i)\right| \tag{10}$$

$$\vec{A} = \left(2 \times T' + P_{grid} \times rand(0,1)\right) - T' \tag{11}$$

$$\vec{C} = rand(0,1) \tag{12}$$

$$P_{grid} = \left|P_{opt} - P_{Itr}(i)\right| \tag{13}$$

$$S = \left(\sqrt{rand(2,3)e^{-\frac{Itr}{rand(1.5,2)}} - e^{-Itr}}\right)^2 \tag{14}$$

4.4 Relocating procedure

In this phase, the position of each emperor penguin is to be modified using the calculated distance (D) as found in equation (15).

$$P_{Itr+1} = P_{Itr} + \vec{A}D \tag{15}$$

Now the pseudo code of the basic algorithm developed by Dhiman and Kumar [15] can be summarized as follows:

```
$Input\ the\ Populatin\ size\ (N), MaxItr, and\ R\ parameters$
$Generate\ the\ initial\ population$
$Evaluate\ each\ solution\ in\ population\ and\ store\ the\ best\ solution\ (P_{opt})$
$Itr = 1$
$While\ Itr \le MaxItr\ do:$
    $i = 1$
    $While\ i \le N\ do:$
        $T' = T - \frac{MaxItr}{Itr - MaxItr}$
        $A = M \times \left(T' + |P_{opt} - P_{Itr}(i)| \times rand(0,1)\right) - T'$
        $S = \left(\sqrt{rand(2,3)e^{-\frac{Itr}{rand(1.5,2)}} - e^{-Itr}}\right)^2$
        $D = |S\ .\ P_{Itr}(i) - rand\ P_{opt}|$
        $P_{Itr+1}(i) = P_{Itr}(i) - \vec{A}D$
        $if\ f(P_{Itr+1}(i)) \le f(P_{opt})\ then:$
                $P_{opt} = P_{Itr+1}(i)$
        $i = i + 1$
    $Itr = Itr + 1$
$Return\ P_{opt}$
```

5 Modified penguin optimizer and comparative results

This section presents a modification of the EPO algorithm to adapt it for solving FCTP. The new modification of the algorithm considers adding an information vector ($P_{IV}(i)$)[16]. Such information vector will be created during the relocating procedure of the algorithm. The creation of this vector is done using the positions of two emperor penguins. The first position is associated with the position of penguin (i) at iteration (Itr) in the population ($P_{Itr}(i)$), while the second position is associated with the relocated position of that penguin ($P_{Itr+1}(i)$). The threshold herein is used as a predetermined number from the interval [0,1]. It is used to determine whether to select a component from the $P_{Itr}(i)$ or from $P_{Itr+1}(i)$. So, the steps to create the information vector can be summarized as follows:

```
$j = 1$
$While\ j \le \dim(P_{Itr}(i))\ do:$
    $if\ rand > threshold:$
        $P_{IV}(i)[j] = P_{Itr+1}(i)[j]$
    $else:$
        $P_{IV}(i)[j] = P_{Itr}(i)[j]$
    $j = j + 1$
```

The created $P_{IV}(i)$ replaces the $P_{Itr}(i)$ if its fitness value is better. Now, the modified version of the emperor penguin optimizer can be summarized as follows:

```
$Input\ the\ Populatin\ size\ (N), MaxItr, and\ R\ parameters$
$Generate\ the\ initial\ population$
```

```
$Evaluate\ each\ solution\ in\ population\ and\ store$
$the\ best\ solution\ (P_{opt})$
$Itr = 1$
$While\ Itr \leq MaxItr\ do:$
    $i = 1$
    $While\ i \leq N\ do:$
        $TA = T - \frac{MaxItr}{Itr - MaxItr}$
        $A = M \times (TA + |P_{opt} - P_i| \times rand) - TA$
        $S = \left(\sqrt{f \,.\, e^{-Itr/l} - e^{-Itr}}\right)^2$
        $D = |S \,.\, P_{Itr}(i) - rand\ P_{opt}|$
        $P_{Itr+1}(i) = P_{Itr}(i) - A \,.\, D$
        $j = 1$
        $While\ j \leq \dim(P_{Itr}(i))\ do:$
            $if\ rand > threshold:$
                $P_{IV}(i)[j] = P_{Itr+1}(i)[j]$
            $else:$
                $P_{IV}(i)[j] = P_{Itr}(i)[j]$
            $j = j + 1$
        $if\ f(P_{Itr+1}(i)) \leq f(P_{opt})\ then:$
            $P_{opt} = P_{Itr+1}(i)$
        $i = i + 1$
    $Itr = Itr + 1$
$Return\ P_{opt}$
```

The information vector proves efficiency in the proposed modification of the EPO algorithm that prevents stagnation in local optima, especially in the multi-model test optimization functions. EPO mainly uses a vector-based methodology to deal with positions and modify them using its relocating procedure and the new proposed information vector creation process. In order to test the performance of the algorithm, 19 test optimization problems are selected from https://www.sfu.ca/~ssurjano/optimization.html to be solved using the EPO algorithm.

Figure 1 and Figure 2 show the 3D plots of the 19 optimization functions. To evaluate the effectiveness of the modified EPO, a comparison is done with the classical EPO and particle swarm optimization (PSO) algorithms. These algorithms are implemented in Python, and the comparisons are conducted on a PC featuring a core-i5 3.40 GHz CPU and 4 GB of memory. Table 3 shows the comparative results, where the highlighted values in the table prove the effectiveness of the modified EPO in terms of objective values and robustness. Furthermore, the Friedman test [17] is applied to prove that the null hypothesis is rejected, since the p-value for means is 0.00093, and for standard deviations is 0.00064. The comparative results show that the modified EPO outperforms the other algorithms in 15 problems in terms of means and standard deviations.

Table 2: Test optimization problems

No.	*Function Name*	$f(x)$	*Global Minimum*
1	*Ackley*	$20\left(e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}}\right) - \left(e^{\frac{1}{d}\sum_{i=1}^{d}cos(2\pi x_i)}\right) + 20 + e, x_i \in [-33, 33]$	$f(x^*) = 0, x^* = (0, \dots, 0)$
2	*Bohachevsky*	${x_1}^2 + 2{x_2}^2 - 0.3\,cos(31\pi x_1) - 0.4\,cos(31\pi x_2) + 0.7, x_i \in [-100, 100]$	$f(x^*) = 0, x^* = (0, 0)$
3	*Booth*	$(x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2, \quad x_i \in [-10, 10]$	$f(x^*) = 0, x^* = (1, 3)$
4	*Bukin*	$100\sqrt{\lvert x_2 - 0.01{x_1}^2\rvert} + 0.01\lvert x_1 + 10\rvert, \quad x_i \in [-15, 3]$	$f(x^*) = 0, x^* = (-10, 0)$
5	*Cross-in-Tray*	$-0.0001\left(\left\lvert sin(x_1)\,sin(x_2)e^{\left\lvert 100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert}\right\rvert + 1\right)^{0.1}, x_i \in [-15,15]$	$f(x^*) = -2.06261, x^* = (1.3491, -1.3491$
6	*Drop Wave*	$-\frac{1 + cos\left(12\sqrt{x_1^2 + x_2^2}\right)}{0.5(x_1^2 + x_2^2) + 2}, x_i \in [-5.12, 5.12]$	$f(x^*) = -1.5, x^* = (0,0)$

No.	*Function Name*	$f(x)$	*Global Minimum*
7	*Discus*	$10^6 x_1^2 + \sum_{i=2}^{D} x_i^2 \, , x_i \in [0, 100]$	$f(x^*) = 0, x^* = (0, \dots, 0)$
8	*Easom*	$-\cos(x_1)\cos(x_2)\, e^{(-(x_1-\pi)^2-(x_2-\pi)^2)}, \quad x_i \in [-100, 100]$	$f(x^*) = -1, x^* = (\pi, \pi)$
9	*Eggholder*	$-(x_2+47)\sin\left(\sqrt{\left\lvert x_2 + \frac{x_1}{2} + 47\right\rvert}\right) - x_1 \sin\left(\sqrt{\lvert x_1 - (x_2+47)\rvert}\right), x \in [-500, 500]$	$f(x^*) = -959.6407, x^* = (512, 404.2319)$
10	*Griewank*	$\sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \, , x \in [-600, 600]$	$f(x^*) = 0, x^* = (0, \dots, 0)$
11	*Holder Table*	$-\left\lvert \sin(x_1)\cos(x_2) e^{\left(\left\lvert 1 - \frac{\sqrt{x_1^2+x_2^2}}{\pi} \right\rvert\right)} \right\rvert, x_i \in [-10, 10]$	$f(x^*) = -19.2085, x^* = (8.05502, -9.664$
12	*Michalewicz*	$-\sum_{i=1}^{d} \sin(x_i)\sin^{20}\left(\frac{i x_i^2}{\pi}\right), x \in [0, \pi]$	$f(x^*) = -1.8013, x^* = (2.20, 1.57)$
13	*Modified Schwefel*	$418.9829 \times D - \sum_{i=1}^{D} g(z_i) \, , z_i = x_i + 4.209687462275036e+002, x_i \in [-500, 500]$ $g(z_i) = \begin{cases} z_i \sin\left(\lvert z\rvert^{\frac{1}{2}}\right), & if \ \lvert z_i\rvert \le 500 \\ \left(500 - mod(z_i, 500)\right)\sin\left(\sqrt{500 - \lvert mod(z_i, 500)\rvert}\right) - \frac{(z_i}{1} \\ (mod(\lvert z_i\rvert, 500) - 500)\sin\left(\sqrt{\lvert mod(\lvert z_i\rvert, 500) - 500\rvert}\right) - (\end{cases}$	$f(x^*) = 0, x^* = [0, \dots, 0]$
14	*Rastrigin*	$10d + \sum_{i=1}^{d} [x^2 - 10\cos(2\pi\, x_i)] \, , \quad x_i \in [-5, 5]$	$f(x^*) = 0, x^* = [0, \dots, 0]$
15	*Rosenbrock*	$\sum_{i=1}^{d-1} [100(x_{i+1} - {x_i}^2)^2 + (x_i - 1)^2] \, , \quad x_i \in [-5, 10]$	$f(x^*) = 0, x^* = (1, \dots, 1)$
16	*Schwefel*	$418.9829d - \sum_{i=1}^{d} x_i \sin\left(\sqrt{\lvert x_i\rvert}\right), \quad x_i \in [-500, 500]$	$f(x^*) = 0, x^* = (420.9687, \dots, 42$

No.	*Function Name*	$f(x)$	*Global Minimum*
17	*six-hump*	$\left(4-2.1x_1^2+\frac{x_1^4}{3}\right)x_1^2+x_1x_2+(-4+4x_2^2)x_2^2,\ x_i\in[-3,3]$	$f(x^*) = -1.0316, x^* = (0.0898, -0.712$
18	*Sphere*	$\sum_{i=1}^{n} x_i^2, x_i \in [-5,5]$	$f(x^*)=0, x^* = [0,\dots,0]$
19	*Zakharov*	$\sum_{i=1}^{d} x_i^2+\left(\sum_{i=1}^{d} 0.5ix_i\right)^2+\left(\sum_{i=1}^{d} 0.5ix_i\right)^4, \quad x_i\in[-5,10]$	$f(x^*)=0, x^* = (0,\dots,0)$

Table 3: The comparative results of the modified EPO with the classical EPO and PSO in 19 test optimization problems

Functions	EPO mean	EPO Classical mean	PSO mean	EPO std	EPO Classical std	PSO std
Ackley	***3.30***	7.71	9.64	***0.62***	1.14	2.43
Bohachevsky	0.22	0.23	***0.01***	0.20	0.21	***0.02***
Booth	0.00	0.00	***0.00***	0.00	0.00	***0.00***
Bukin	0.89	1.62	***0.29***	0.44	0.57	***0.22***
Cross-in-Tray	***-2.06***	***-2.06***	***-2.06***	***0.00***	***0.00***	***0.00***
Drop Wave	***-1.00***	-0.97	-0.99	***0.00***	0.03	0.03
Discus	***101.99***	313.97	424.81	***22.33***	75.34	38.10
Easom	-0.98	-0.59	***-1.00***	***0.00***	0.30	0.00
Eggholder	***-942.86***	-933.63	-915.82	***10.38***	17.30	35.44
Griewank	***1.48***	35.68	6.07	***0.24***	15.66	1.96
Holder Table	***-19.21***	-19.18	-19.21	***0.00***	0.01	0.01
Michalewicz	***-8.29***	-5.92	-5.12	***0.22***	0.52	0.65
Modified Schwefel	***73.83***	1574.72	1135.02	***24.10***	64.48	71.17
Rastrigin	***6.59***	25.61	40.20	***2.68***	7.23	9.12
Rosenbrock	***11.81***	49.92	2811.51	***1.75***	27.63	2926.26
Schwefel	***263.11***	1953.86	2114.75	142.06	***109.59***	338.20
six-hump	***-1.03***	-1.03	-1.03	***0.00***	0.00	0.00
Sphere	***0.01***	0.01	1.70	***0.00***	0.00	0.79
Zakharov	***2.58***	11.62	50.02	***1.27***	6.37	24.62

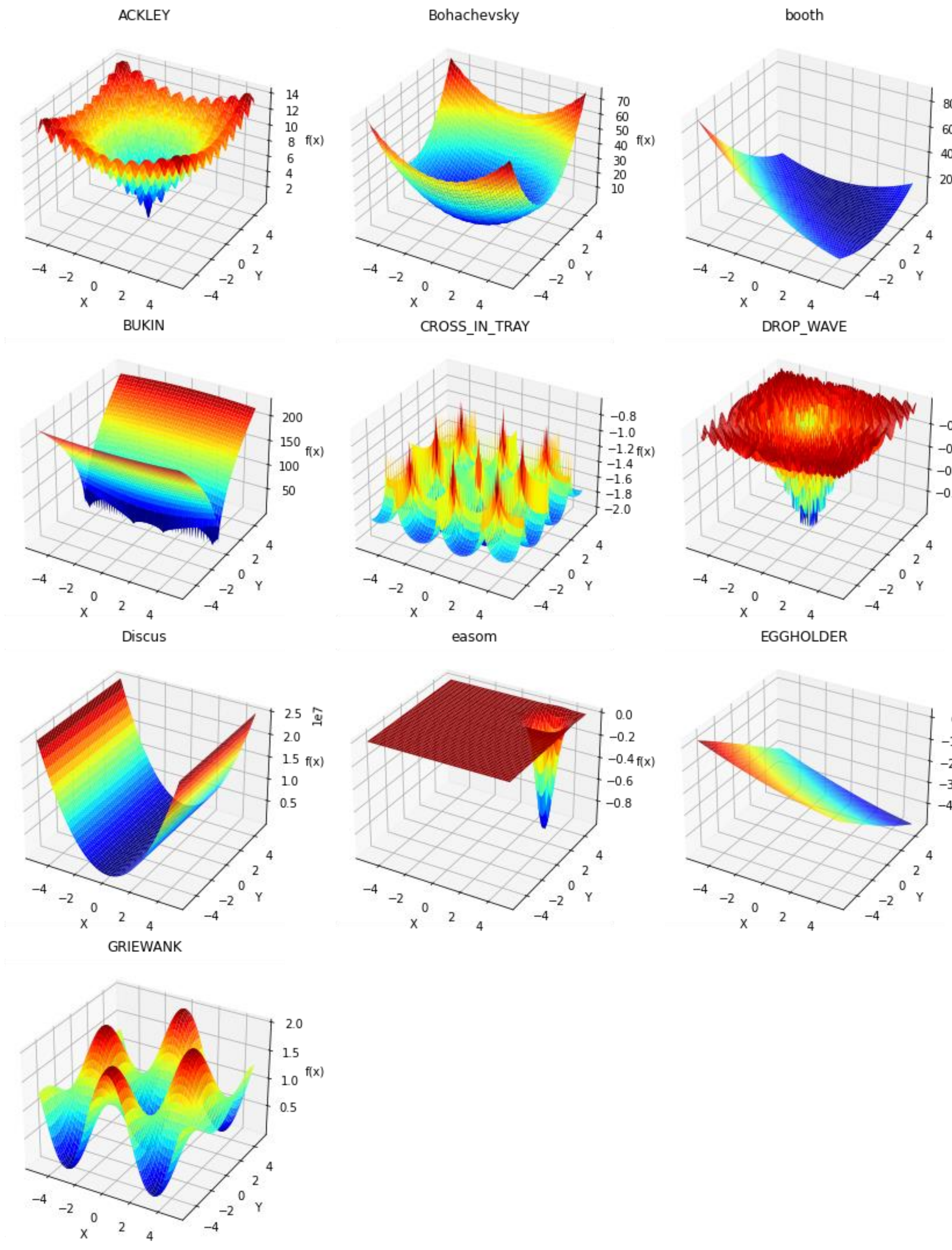

Figure 1: The plots of the first 10 optimization functions

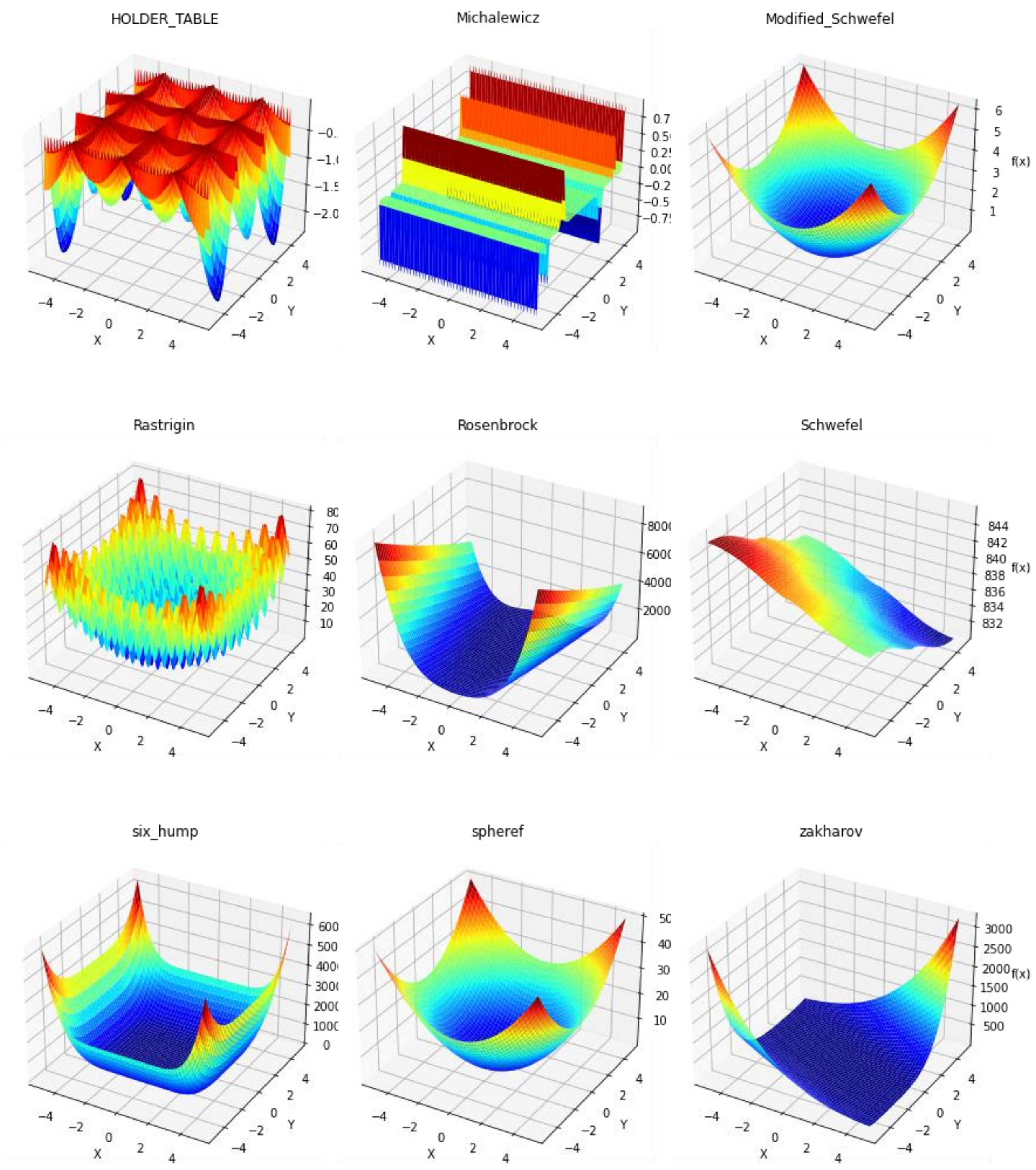

Figure 2: The plots of the 11 to 19 optimization functions

6 Modified EPO for solving FCTP

To adapt EPO for solving FCTP, a priority rule that uses a weighted vector is developed in this paper. The length of the weighted vector equals the ordered product which consists of the set of all ordered pairs of supplies and demands. The transshipment problem involves transient nodes that can be considered for both supplies and demands simultaneously. Hence, the number of supply nodes (SN) is equal to the sum of the number of supplies and the number of transient nodes, while the number of demand nodes (DN) is equal to the sum of the number of demands and the number of transient nodes. So, the number of ordered pairs in our case herein can be calculated using equation (16):

$$Number\ of\ odered\ pairs = SN \times DN \qquad (16)$$

Each ordered pair consists of two components. The first component is the supply node number, and the second component is the demand node number. By arranging these ordered pairs and solving the problem according to this arrangement, a heuristic solution can be obtained. As aforementioned, a proposed weighted vector is used to generate a priority rule for these ordered pairs, where each ordered pair has an associated weight, and the highest weighted ordered pair will be assigned first. Table 4 shows an example of the ordered product for a problem where the number of supplies equals three and the number of demands equals four.

Table 4: An example of an ordered pair for a problem of three supplies and four demands

Supplies	Demands	Ordered Pairs
1	1	(1,1)
1	2	(1,2)
1	3	(1,3)
1	4	(1,4)
2	1	(2,1)
2	2	(2,2)
2	3	(2,3)
2	4	(2,4)
3	1	(3,1)
3	2	(3,2)
3	3	(3,3)
3	4	(3,4)

In the heuristic procedure, the arrangement of ordered pairs is to be arranged according to the weighted vector, which can be initially generated randomly for each ordered pair. For illustration, Table 5 shows an example of arranging these ordered pairs according to a weighted vector.

Table 5: An example of using the weighted vector to rearrange the problem’s ordered pairs.

Ordered Pairs	Weights
(3,3)	0.99
(2,3)	0.90
(1,1)	0.83
(1,3)	0.71
(3,2)	0.44
(1,2)	0.43
(3,1)	0.43
(2,1)	0.37
(3,4)	0.10
(2,2)	0.05
(2,4)	0.05
(1,4)	0.00

The proposed heuristic procedure of the algorithm can be implemented using the arranged ordered pairs, where it considers assigning the quantities of the problem according to the arrangement found by the weighted vector. The heuristic procedure now can be illustrated using the following pseudo code:

```
$OP_{arranged}$
$= the\ arragned\ ordred\ pairs\ with\ respect$
$to\ a\ weighted\ vector$
$Set\ S = \{s_i | i \in I\}\ and\ D = \{d_j | j \in J\}$
$Create\ Solution\ matrix\ that\ is\ initialized\ by\ zeros$
$SN\ rows\ and\ DN\ columns$
$Index = 0$
$While\ OP_{arranged}\ is\ not\ empty\ do:$
    $(a, b) = OP_{arranged}[Index]$
    $If\ s_a = d_b\ then:$
        $Solution(a, b) = s_a$
        $s_a = 0$
        $d_b = 0$
        $OP_{arranged}$
        $= \{(i, j) | i \in I\ and\ i \neq a, j \in J\ and\ j \neq b\}$
    $End\ if$
    $If\ s_a < d_b\ then:$
        $Solution(a, b) = s_a$
        $d_b = d_b - s_a$
        $s_a = 0$
        $OP_{arranged} = \{(i, j) | i \in I\ and\ i \neq a, j \in J\}$
    $End\ if$
    $If\ s_a > d_b\ then:$
        $Solution(a, b) = d_b$
        $s_a = s_a - d_b$
        $d_b = 0$
        $OP_{arranged} = \{(i, j) | i \in I, j \in J\ and\ j \neq b\}$
    $End\ if$
$End\ while$
$Return\ total\ cost\ and\ solution\ matrix$
```

The proposed modified Penguin Algorithm can now solve the problem by utilizing the previously mentioned heuristic procedure and a weighted vector. These weighted vectors represent the positions of the penguins. By modifying the weighted vector, new solutions can be obtained using the heuristic procedure. The heuristic procedure can now serve as the optimization function that needs to be optimized, with the weighted vectors representing the positions of the penguins.

6.1 Computational complexity

The algorithm initializes with a number of priority vectors equal to N solutions with d dimensions. So, the initialization process requires $O(N \times d)$. The heuristic procedure step count is less than d, since not all of the ordered pairs are selected in the heuristic procedure. Thus, the heuristic procedure requires $O(N \times d \times MaxItr)$, hence it will repeat until the maximum number of iterations is reached. The step count of the rest of the functions required for calculating new positions in the EPO equals N with k formulas. So, it this requires $O(N \times k)$. The total time complexity required for the algorithm is $O(N \times d \times MaxItr \times k)$ and the space complexity is $O(N \times d)$, since the algorithm only works with population that initialized by N solutions with d dimensions.

6.2 Experimental design

In order to obtain the optimal settings of the algorithm, an experimental design is done on both the hybrid EPO and PSO algorithms for solving the fixed charged transshipment problem. The code of the hybrid algorithms and the other codes related to the problem are coded using python and can be found in https://github.com /MZakaraia/EPO_Transshipment/. The selected problems for experimental design are generated using the generate problem's function found in previously mentioned GitHub repository. The modified EPO algorithm has 3 parameters, which are $MaxItr$, the population size ($PopSize$), and $Radius$. For each parameter, 4 levels are chosen as shown in Table 6.

The full factorial design requires $4^3 \times 5 = 320$ trails for 5 replicates. This number of experiments can be reduced using Taguchi's orthogonal arrays. In order to select the convenient orthogonal array, the degrees of freedom should be calculated. So, the degree of freedom for such experiment is 1 of the overall mean and 3 for each parameter, which means the total degrees of freedom herein is 10. The most convenient orthogonal array for this experiment is $L_{16}(4^3)$. The 16 runs of the experimental design are implemented each 5 times to calculate the signal to noise ratio (SNR) using equation (17) after normalizing the outputs. Figure 3shows the optimized parameter levels for each parameter, which is 20 iterations, 20 penguins, and the radius should be equal 2.

Table 6: Parameter levels for the modified EPO algorithm

$MaxItr$	$PopSize$	$Radius$
20	20	0
50	50	-1
70	70	1
100	100	2

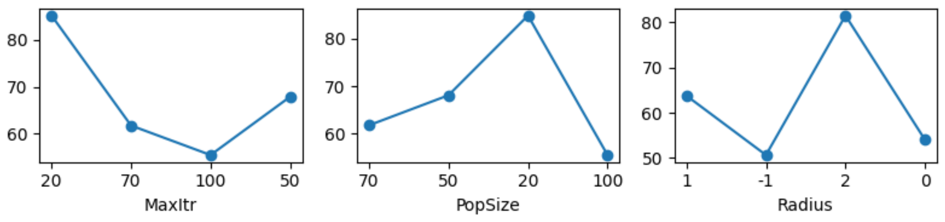

Figure 3: The main effect plots of signal to ratio of EPO

$$SNR = 10\log\left(\frac{\mu^2}{\sigma^2}\right) \quad (17)$$

For the hybridized PSO algorithm, there are 5 parameters, which are $MaxItr$, $PopSize$, Inertia weight, personal weight, and the global weight. The proposed levels for each parameter are found in Table 7.

Table 7: Parameter levels for the PSO algorithm

$MaxItr$	$PopSize$	*Inertia weight*	*personal weight*	*global weight*
20	20	0.1	0.1	0.1
50	50	0.3	0.3	0.3
70	70	0.5	0.5	0.5
100	100	0.6	0.6	0.6

The required number of trails for the full factorial design for the hybrid PSO according to the levels in Table 7 is $4^5 \times 5 = 5120$ trails. The Taguchi's orthogonal arrays again can be used to reduce this number using the $L_{16}(4^5)$, where the number of trails is 80 trails for 5 replicates. Figure 4 shows the optimized parameter levels for the hybrid PSO algorithm, which are 20 iterations, 20 particles, 0.1 for the inertia weight, 0.6 for personal weight, and 0.6 for global weight. The optimized parameter levels are to be used in the computational results section to show the comparative results between the EPO and PSO algorithms.

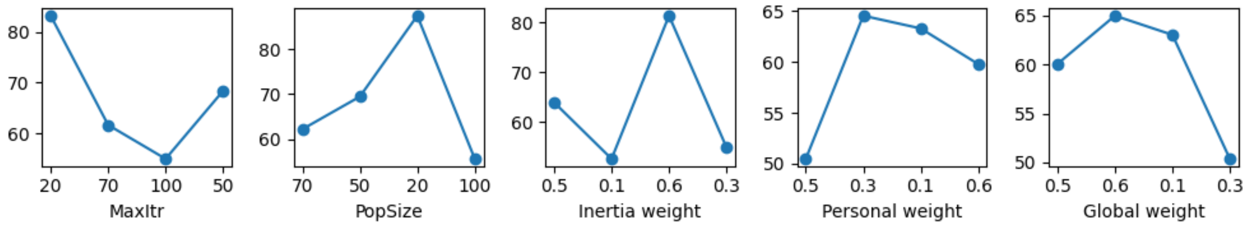

Figure 4: The main effect plots of signal to ratio of PSO

6.3 Computational results

This section presents the implementation of the modified EPO algorithm for solving 30 generated problems, which can be found at https://github.com/MZakaraia/EPO_Ttransshipment. The problem sizes cover three different forms of transshipment problem sizes: $3 \times 3 \times 2$, $4 \times 4 \times 3$, and $5 \times 5 \times 4$. All these problems are generated using the generate problems function in the Transshipment.py file. This function allows for generating more fixed-charge transshipment problems with different sizes. Therefore, the included problems are considered benchmarks for future comparisons.

Both the EPO and PSO algorithms were implemented to solve the 30 problems using the optimized parameter levels found through the experimental design. The convergence curves for the problems are shown in Figure 5, 6, and 7. Table 8 presents the comparative results. The Wilcoxon test [18] was performed on selected metrics (mean, standard deviation, maximum, minimum) between the EPO and PSO results. The p-value for each metric indicates rejection of the null hypothesis since the p-value for means is 1.89×10^{-9} and for standard deviations is 0.00046. Therefore, the comparative results in Table 8 conclude that the proposed EPO algorithm outperforms the PSO algorithm in terms of mean results and robustness for solving fixed-charge transshipment problems.

Table 8: The comparative results of the 30 fixed charged transshipment problems between the modified EPO and PSO

Problem	EPO Mean	PSO Mean	EPO Std	PSO Std	EPO Max	PSO Max	EPO Min	PSO Min
3X3X2_0	***41797***	42206.3	***92.29084462***	508.4504007	***42031***	43136	***41719***	***41719***
3X3X2_1	***40680.5***	41229.8	***267.1895395***	567.3982376	***41237***	42530	***40547***	***40547***
3X3X2_2	***33693.5***	34348.6	***88.86534758***	572.6377913	***33960***	35163	***33663***	***33663***
3X3X2_3	***29758.4***	30133	***144.9497844***	424.4848643	***30117***	30874	***29688***	***29688***
3X3X2_4	***42124.7***	42336.3	***153.5689096***	240.9514681	***42445***	42713	***42030***	***42030***
3X3X2_5	***37551***	37904.9	***184.3805847***	383.2465134	***37908***	38456	***37432***	***37432***
3X3X2_6	***38157***	38436.7	***23.37947818***	247.5019394	***38198***	38968	***38142***	38180
3X3X2_7	***33565***	34001.8	***0***	411.3178333	***33565***	34920	***33565***	***33565***
3X3X2_8	***30994***	31619.5	***0***	645.4463959	***30994***	32779	***30994***	***30994***
3X3X2_9	***31027.1***	31353.8	***102.3***	298.0412052	***31334***	31873	***30993***	***30993***
4X4X3_0	***47553.3***	49048.4	***632.032602***	1151.657953	***49443***	50903	***47324***	***47324***
4X4X3_1	***58361.5***	59400.7	***128.9234269***	351.1797403	***58663***	60087	***58217***	58880
4X4X3_2	***60169.8***	60771.1	813.7788152	***594.8965372***	***61489***	61697	***59351***	59667
4X4X3_3	***71064.5***	72923.7	1106.743895	***620.5789313***	***73668***	73803	***70269***	71847
4X4X3_4	***65386***	67792.3	***762***	1303.083961	***67672***	71176	***65132***	65740
4X4X3_5	***64932.1***	65240.9	945.1474435	***811.3807306***	***66407***	67011	***63926***	64370
4X4X3_6	***56582.9***	57742.8	***95.05519449***	869.7954702	***56853***	59901	***56538***	56634
4X4X3_7	***64673.8***	65989.4	***242.1808415***	599.2033378	***65260***	66880	***64432***	65080
4X4X3_8	***64801.4***	66226.3	***598.4196187***	632.1170857	***66091***	67494	***64249***	65245
4X4X3_9	***64173.9***	65578	452.4350672	***450.8871256***	***65236***	66499	***63846***	64718
5X5X4_0	***99127***	101002	***525.5029971***	1238.38217	***99858***	103173	***98290***	98948
5X5X4_1	***103558.6***	105635.4	***971.9633944***	1316.379824	***105953***	107749	***102571***	104127
5X5X4_2	***118153.9***	120260	***500.3739502***	1535.788136	***119108***	123992	***117548***	118137
5X5X4_3	***100174.3***	103129.5	***222.7321486***	1607.177977	***100741***	107405	***99905***	100916
5X5X4_4	***105533.9***	107772.6	***563.3280483***	1053.34811	***106665***	110543	***104657***	106575
5X5X4_5	***101425.3***	103906.4	***165.7950844***	1003.615285	***101794***	105193	***101214***	102351
5X5X4_6	***97936.6***	100878.4	***1137.296901***	2000.878567	***100376***	104848	***96986***	97753
5X5X4_7	***106153.2***	107832.2	1274.031224	***757.3083652***	***108707***	108989	***105099***	106449
5X5X4_8	***86597.7***	87972.9	1604.486089	***1036.310229***	***89020***	89773	***84759***	86412
5X5X4_9	***92603.3***	94816.2	***307.2627703***	1282.450217	***93358***	97327	***92113***	92515

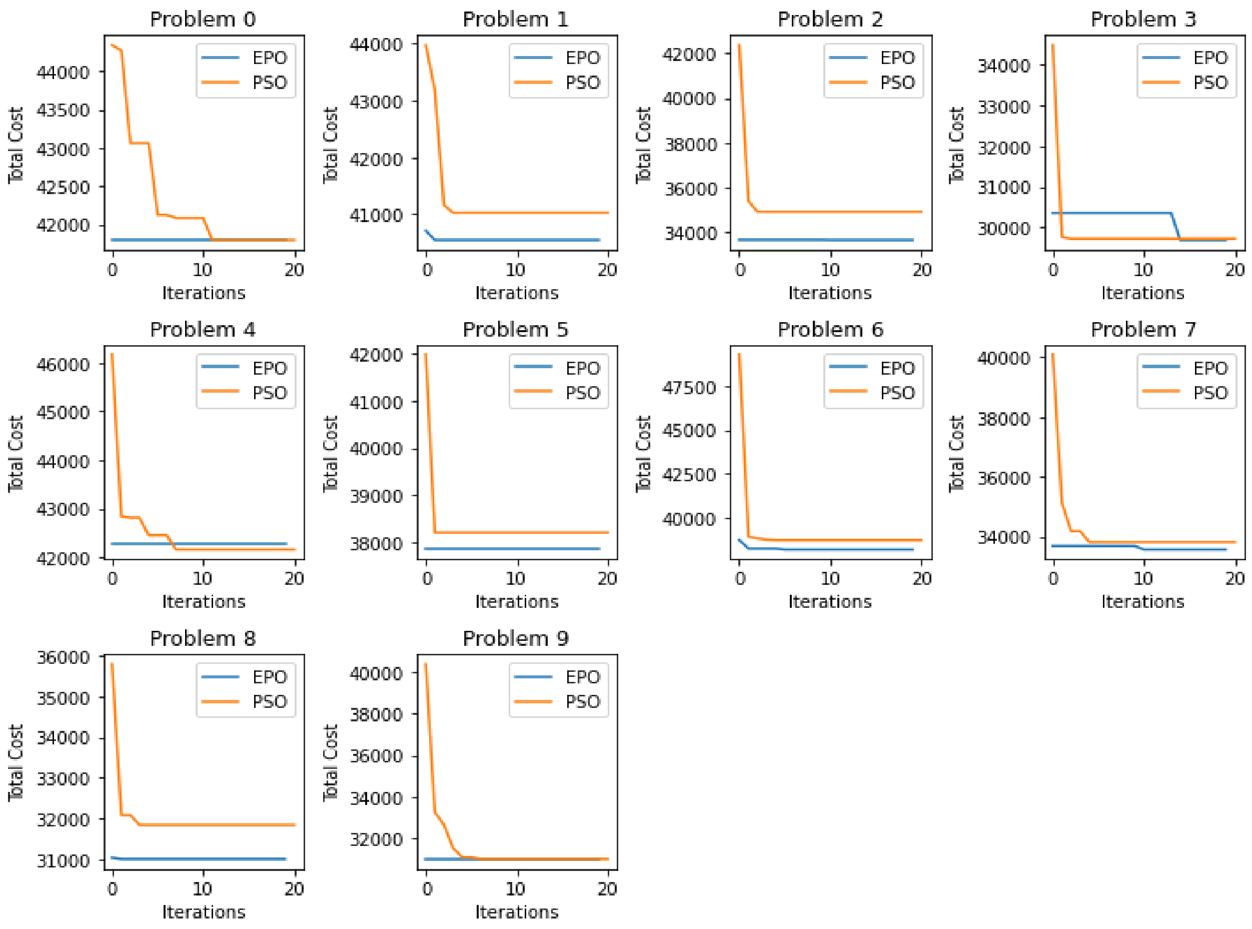

Figure 5: The convergence curve of the 3 × 3 × 2 problems

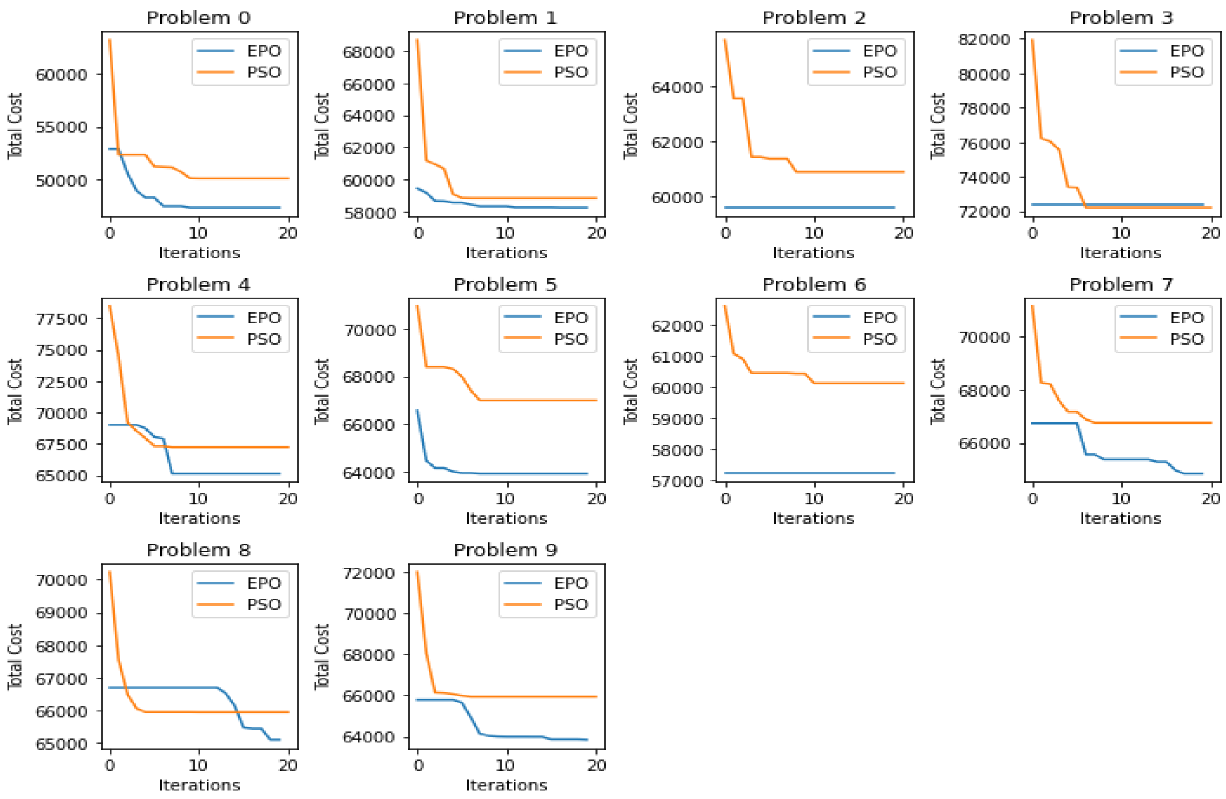

Figure 6: The convergence curve of the 4 × 4 × 3 problems

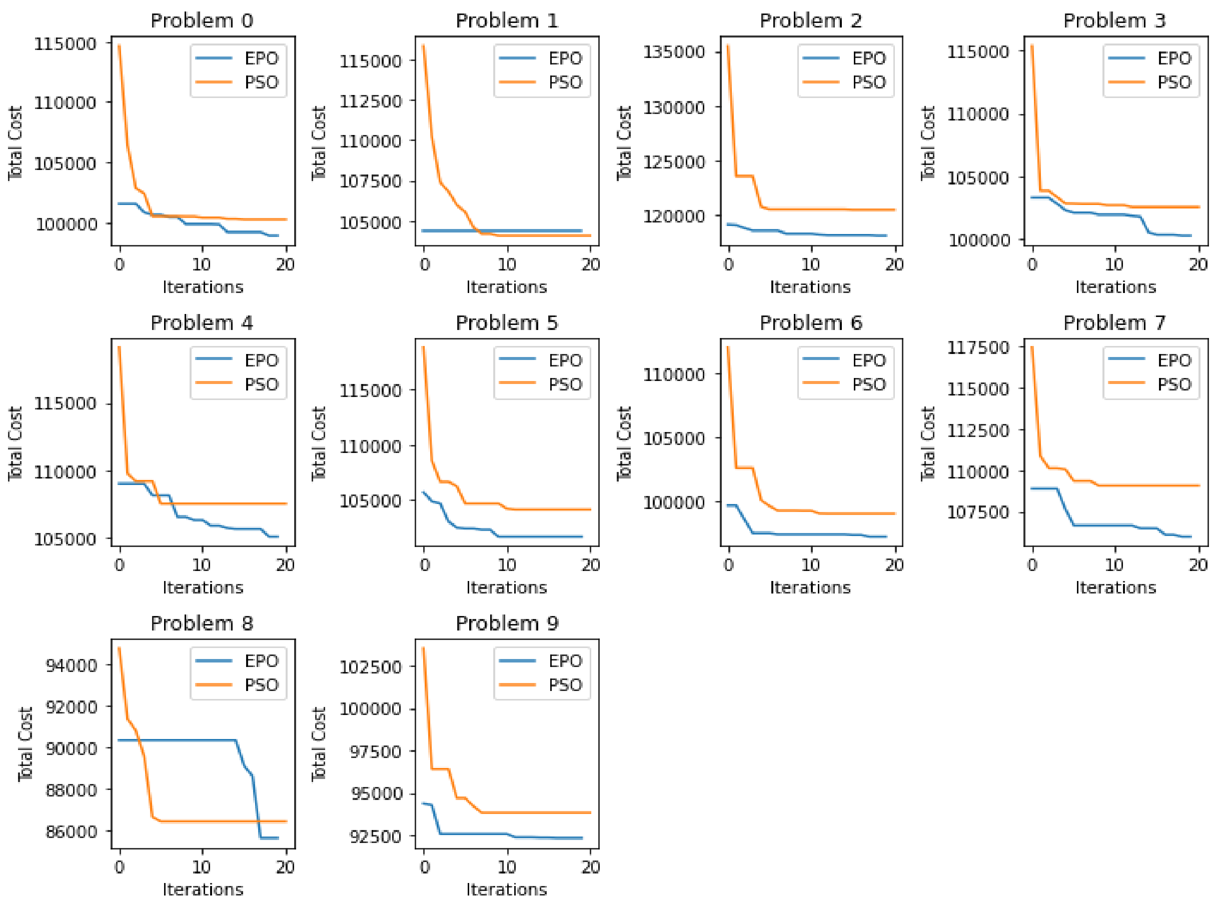

Figure 7: The convergence curve of the 5 × 5 × 4 problems

7 Conclusion

In conclusion, this paper introduced a modified Emperor Penguin algorithm tailored for solving FCTP. The algorithm demonstrated its effectiveness in finding high-quality solutions by utilizing new benchmarks specifically designed for the problem. The computational results presented in this study provide valuable insights into the algorithm's performance. The mean results showcased the algorithm's ability to achieve competitive solutions for the FCTP, while the standard deviation and Relative Standard Deviation offered measures of its robustness. The findings of this research contribute to the field of logistics and supply chain management by offering an optimized algorithmic approach for addressing the FCTP. The modified Emperor Penguin algorithm, with its robustness and improved solution quality, holds great potential for enhancing supply chain operations and optimizing transshipment processes.

Future research directions may involve further fine-tuning of the modified algorithm and expanding the benchmark suite to encompass a wider range of real-world scenarios. Additionally, investigating the algorithm's performance on larger-scale instances and exploring its applicability to other related optimization problems would be beneficial. The future research also may include extending the formulations of the transshipment problem to cover:

- The solid transshipment problem by including constraints related to the type of transportation and products.
- The capacitated fixed charged transshipment by considering capacity constraints related to each transshipment node.

In conclusion, this paper's findings highlight the promising capabilities of the modified Emperor Penguin algorithm for tackling the Fixed Charged Transshipment Problem, providing a valuable tool for optimizing supply chain operations and fostering efficiency in logistics management.

Acknowledgement

This work was funded by the University of Jeddah, Jeddah, Saudi Arabia, under grant No. (UJ-02-068-DR). The authors, therefore, acknowledge with thanks the University of Jeddah technical and financial support.

References

[1] Y. T. Herer and M. Tzur, “The dynamic transshipment problem,” *Nav. Res. Logist.*, vol. 48, no. 5, pp. 386–408, 2001, doi: 10.1002/nav.1025.

[2] P. M. Reyes, “Logistics networks: A game theory application for solving the transshipment problem,” *Appl. Math. Comput.*, vol. 168, no. 2, pp. 1419–1431,

2005, doi: 10.1016/j.amc.2004.10.030.

[3] Y. T. Herer, M. Tzur, and E. Yücesan, "The multilocation transshipment problem," *IIE Trans. (Institute Ind. Eng.*, vol. 38, no. 3, pp. 185–200, 2006, doi: 10.1080/07408170500434539.

[4] N. Belgasmi, L. Ben Saïd, and K. Ghédira, "Evolutionary multiobjective optimization of the multi-location transshipment problem," *Oper. Res.*, vol. 8, no. 2, pp. 167–183, 2008, doi: 10.1007/s12351-008-0015-5.

[5] D. K. Sharma and R. K. Jana, "A hybrid genetic algorithm model for transshipment management decisions," *Int. J. Prod. Econ.*, vol. 122, no. 2, pp. 703–713, 2009, doi: 10.1016/j.ijpe.2009.06.036.

[6] A. Khurana and S. R. Arora, "Solving transshipment problems with mixed constraints," *Int. J. Manag. Sci. Eng. Manag.*, vol. 6, no. 4, pp. 292–297, 2011, doi: 10.1080/17509653.2011.10671176.

[7] D. Özdemir, E. Yücesan, and Y. T. Herer, "Multi-location transshipment problem with capacitated production," *Eur. J. Oper. Res.*, vol. 226, no. 3, pp. 425–435, 2013, doi: 10.1016/j.ejor.2012.11.014.

[8] A. Khurana, T. Verma, and S. R. Arora, "Solving time minimising transshipment problem," *Int. J. Shipp. Transp. Logist.*, vol. 7, no. 2, pp. 137–155, 2015, doi: 10.1504/IJSTL.2015.067848.

[9] A. Kumar, R. Chopra, and R. R. Saxena, "An Efficient Algorithm to Solve Transshipment Problem in Uncertain Environment," *Int. J. Fuzzy Syst.*, vol. 22, no. 8, pp. 2613–2624, 2020, doi: 10.1007/s40815-020-00923-9.

[10] H. Garg, A. Mahmoodirad, and S. Niroomand, "Fractional two-stage transshipment problem under uncertainty: application of the extension principle approach," *Complex Intell. Syst.*, vol. 7, no. 2, pp. 807–822, 2021, doi: 10.1007/s40747-020-00236-2.

[11] H. Zhao and A. Sharma, "Logistics Distribution Route Optimization Based on Improved Particle Swarm Optimization," *Inform.*, vol. 47, no. 2, pp. 243–252, 2023, doi: 10.31449/inf.v47i2.4011.

[12] M. Khetatba and R. Boudour, "A modified binary firefly algorithm to solve hardware/software partitioning problem," *Inform.*, vol. 45, no. 7, pp. 1–12, 2021, doi: 10.31449/inf.v45i7.3408.

[13] A. A. Almazroi and M. M. Eltoukhy, "Grey Wolf-Based Method for an Implicit Authentication of Smartphone Users," *Comput. Mater. Contin.*, vol. 75, no. 2, pp. 3729–3741, 2023, doi: 10.32604/cmc.2023.036020.

[14] M. F. Mohamed, M. M. Eltoukhy, K. Al Ruqeishi, and A. Salah, "An Adapted Multi-Objective Genetic Algorithm for Healthcare Supplier Selection Decision," *Mathematics*, vol. 11, no. 6, 2023, doi: 10.3390/math11061537.

[15] G. Dhiman and V. Kumar, "Emperor penguin optimizer: A bio-inspired algorithm for engineering problems," *Knowledge-Based Syst.*, vol. 159, pp. 20–50, 2018, doi: 10.1016/j.knosys.2018.06.001.

[16] A. Serag, H. Zaher, N. Ragaa, and H. Sayed, "A Modified Emperor Penguin Algorithm for Solving Stagnation in Multi-Model Functions," *Inform.*, vol. 47, no. 10, pp. 71–78, 2023, doi: 10.31449/inf.v47i10.5273.

[17] H. Kaur and A. Kaur, "An Empirical Study of Aging Related Bug Prediction Using Cross Project in Cloud Oriented Software," *Inform.*, vol. 46, no. 8, pp. 105–120, 2022, doi: 10.31449/inf.v46i8.4197.

[18] I. Salman and J. Vomlel, "Learning the Structure of Bayesian Networks from Incomplete Data Using a Mixture Model," *Inform.*, vol. 47, no. 1, pp. 81–94, 2023, doi: 10.31449/inf.v47i1.4497.