



American Journal of Interdisciplinary Research and Innovation (AJIRI)

ISSN: 2833-2237 (ONLINE)

VOLUME 5 ISSUE 4 (2026)

PUBLISHED BY
E-PALLI PUBLISHERS, DELAWARE, USA

The Modified Hybrid Firefly Approach to the Two-Stage Fixed-Charge Transportation Problem

E.M.D.B Ekanayake^{1*}, W.N.P Rodrigo²

Article Information

Received: July 08, 2026

Accepted: September 19, 2026

Published: October 08, 2026

Keywords

Firefly Algorithm, Mixed-Integer Linear Programming, Optimum Solution, Two-Stage Fixed-Charge Transportation Problem

ABSTRACT

The two-stage fixed-charge transportation problem (TS-FCTP) is a transportation problem in supply chain management. This problem considers determining the most cost-effective way to transport goods through an intermediate state, taking into account both the variable transportation costs and the fixed costs associated with opening or using different routes. Various methods have been used to obtain solutions in various ways, either heuristically or exact solutions. The aim of this research paper is to propose a more accurate algorithm for this purpose. TS-FCTP is an optimization problem of the mixed-integer linear programming (MILP) model. Accordingly, the proposed algorithm is presented as a hybrid based on MILP and the modified firefly model. Furthermore, the solution is obtained in Python to improve the computational efficiency. The robustness of the proposed algorithm is tested on twenty problems with node sizes ranging from seven to fourteen. The convergence behavior of the algorithm shows that the model reaches optimal solutions in reasonable computational time. This performance suggests that combining metaheuristics inspired by nature with mathematical programming can provide efficient solutions. Also, as an extension of this research, the goal is to develop a more accurate algorithm for large-scale TS-FCTP problems.

INTRODUCTION

Transportation problems play an important role in supply chain management and distribution management. One of the most widely studied transportation problems is the fixed-charge transportation problem (FCTP), where the variable shipping cost is proportional to the quantity transported, in addition to the fixed cost associated with using a transportation route.

In real-world logistics distribution problems, they are transported through each stage. Accordingly, goods are transported from the production plants to intermediate warehouses and then distributed to the final customers. This distribution structure creates a two-stage fixed-charge transportation problem (TS-FCTP) (Cosma *et al.*, 2019). Thus, this problem becomes more complex because it has to be determined by which routes can be used to distribute how many goods at each stage to achieve the minimum total transportation cost.

TS-FCTP is a MILP model. Although the MILP formulation guarantees optimal solutions, it is difficult to apply it in practice on the large scale of the problem in terms of computational time. To address this challenge, researchers have increasingly explored hybrid approaches that combine specific optimization techniques with metaheuristic algorithms.

In this study, a hybrid optimization framework is proposed to solve the TS-FCTP problem by combining the Firefly algorithm with a MILP solver. In the proposed method, the Firefly algorithm is used to explore different cost transformation parameters that balance the relative

importance of the transportation costs of the two stages of the distribution network. For each candidate parameter generated by the Firefly algorithm, the MILP model is solved to determine the better shipping allocations while satisfying the supply, demand, and flow conservation constraints. This hybrid structure has enhanced the exploration capability of the Firefly algorithm to successful solutions.

LITERATURE REVIEW

Several researchers have studied the problem of fixed charge transport using various exact, heuristic, and metaheuristic approaches. For example, the authors of (Gen *et al.*, 2006) proposed an approach based on genetic algorithms. Later, genetic algorithms were also used to solve this problem. (Jawahar & Balaji, 2009) Related transportation optimization problems were also investigated. Later, how to change the evaluation algorithm to minimize the total cost at each stage has been studied in the literature (Calvete *et al.*, 2016). In addition to the above studies, solving problems with more variables is also considered as an extension of this problem. For this purpose, problems are generated randomly and successfully solved (Cosma *et al.*, 2020). In addition, (Pop *et al.*, 2017) proposed a hybrid model combining genetic algorithm and local search procedures for this purpose. Other methods have been proposed to solve the fixed charge transport problems and address larger cases (Jha *et al.*, 2023). Recent studies have investigated other heuristic and metaheuristic methods to obtain high-quality

¹ Postgraduate Institute of Science, University of Peradeniya, Peradeniya, Sri Lanka

² Faculty of Sciences, Department of Mathematics, University of Peradeniya, Peradeniya, Sri Lanka

* Corresponding author's e-mail: dananjayaekanyake96@gmail.com

solutions to these problems (Thakur, 2025). The results reported in these studies suggest that heuristic approaches can be useful when the computational requirements of fixed charge transport models are difficult to manage with traditional optimization techniques (Jayakumar, 2017).

MATERIALS AND METHODS

In this research study, a modified hybrid algorithm, primarily including the firefly algorithm and MILP, has been proposed.

Firefly algorithm

The assumptions are that all fireflies are unisex, which means that a firefly is attracted to any other firefly in the population. The attraction between fireflies is proportional to their brightness, which means that a less bright firefly is attracted to a brighter one and that it moves randomly when there is no firefly. This attraction decreases as the distance between fireflies increases. Third, the assumption is the function of the optimization problem is modeled as an exponential function of the brightness or light intensity of each firefly (Yang, 2010). The light intensity $I(r)$ of a firefly is given by Where the given medium with fixed light absorption

$$I = I_0 e^{-\gamma r_{ij}^2} \quad (1)$$

coefficient γ its typically varies from 0.1 to 10 and I_0 is the original light intensity when $r_{ij}=0$ (r_{ij} : distance between two fireflies i and j).

The firefly's attractiveness is proportional to the light

$$r_{ij} = \|x_j - x_i\| \quad (2)$$

intensity seen by adjacent fireflies. It defines β .

Where the β_0 denotes the attractiveness at zero distance.

$$\beta = \beta_0 e^{-\gamma r_{ij}^2} \quad (3)$$

The movement of the i^{th} firefly is attracted to another more attractive (brighter) j^{th} firefly is determined by The second terms take into consideration the fireflies'

$$x_{i+1} = x_i + \beta_0 e^{-\gamma r_{ij}^2} (x_j - x_i) + a\mathcal{E} \quad (4)$$

attractiveness, and the third terms is randomization with a , $a \in [0,1]$ which serves as the randomization parameter. $\mathcal{E}$ is a vector of random numbers generated from a gaussian or uniform distribution.

This study's optimization model was created as a MILP. The PuLP optimization library, which offers an interface for defining linear programming and mixed-integer programming models, was used to implement the MILP model in Python (Mitchell, 2011). Visual Studio Code was

used as the development platform for all computational experiments. A Python-based optimization solver that finds the best transportation flow allocation while meeting supply, demand, and flow conservation constraints in TS-FCTP was used to solve the MILP formulation.

Two stage fixed charge transportation problem

The TS-FCTP extends the classical FCTP by introducing the intermediate state of distribution centers between manufactures and customers. The model is formulated as a MILP problem with the following decision variables and parameters (Calvete *et al.*, 2018).

The TS-FCTP extends the classical FCTP by introducing the intermediate state of distribution centers between manufactures and customers. The model is formulated as a MILP problem with the following decision variables and parameters (Calvete *et al.*, 2018).

I, i : set of sources

J, j : set of intermediate distribution centers (DCs)

K, k : set of customers/destinations.

s_i : supply available at source i

d_k : demand required by customer k

b_{ij} : variable transportation cost per unit from source i to DC j .

c_{jk} : variable transportation cost per unit from DC j to customer k .

f_{ij} : fixed charge for opening the source to DC route.

g_{jk} : fixed charge for opening the DC to customer route.

M_{ij} : maximum possible flow on stage-1 route i to j .

M_{jk} : maximum possible flow on stage-2 route j to k .

The decision variables are:

x_{ij} : quantity transported from source i to DC.

y_{jk} : quantity transported from DC j to customer k .

z_{ij} : binary variable that Stage 1 route opening.

w_{jk} : binary variable that stage 2 route opening.

The objective function for the TS-FCTP is to minimize

$$z_{ij} = \begin{cases} 1, & \text{if the route from source } i \text{ to DC } j \text{ is open,} \\ 0, & \text{otherwise} \end{cases}$$

$$w_{jk} = \begin{cases} 1, & \text{if the route from DC } j \text{ to customer } k \text{ is open} \\ 0, & \text{otherwise} \end{cases}$$

the total transportation cost. Accordingly, Mathematical model can be formulated as follows:

$$\text{Minimize } \sum_{i=1}^I \sum_{j=1}^J (x_{ij} b_{ij} + f_{ij} z_{ij}) + \sum_{j=1}^J \sum_{k=1}^K (y_{jk} c_{jk} + g_{jk} w_{jk})$$

Subject to

$$\sum_{j=1}^J x_{ij} \leq s_i \quad i = 1, 2, \dots, I$$

$$\sum_{j=1}^J y_{jk} = d_k \quad k = 1, 2, \dots, K$$

$$\sum_{j=1}^J x_{ij} = \sum_{k=1}^K y_{jk} \quad j = 1, 2, \dots, J$$

$$x_{ij} \leq M_{ij} z_{ij}$$

$$M_{ij} = \min(s_i, \sum_{k \in K} d_k)$$

$$y_{jk} \leq M_{jk} w_{jk}$$

$$M_{jk} = \sum_{k \in K} d_k$$

$$x_{ij} \geq 0$$

$$y_{jk} \geq 0$$

$$z_{ij} \in \{0, 1\}$$

$$w_{jk} \in \{0, 1\}$$

Proposed equations for convert the unit variable cost

The unit variable cost convert function is proposed to create a firefly population with different allocations total transportation cost.

For each firefly λ_n , the variable shipping costs are converted as follow :

$$\text{Stage 1 } b_{ij}^{\lambda_n} = (1 - \lambda_n) b_{ij} \quad (6)$$

$$\text{Stage 2 } c_{jk}^{\lambda_n} = \lambda_n c_{jk} \quad (7)$$

The proposed modified hybrid firefly algorithm, satisfying each of the above parameters and constraints can be describe as follows.

In the proposed algorithm below, at each iteration, every firefly generates a λ value. This λ values are used to modify the unit transportation costs of both stages. The modified costs are then provided as input to the MILP model, which determines the optimal allocation matrices and the corresponding total transportation cost. The obtained cost is treated as the brightness (fitness) value of the firefly and is used to update the population according the firefly algorithm rules.

Proposed algorithm

Step 1: For each firefly $n=1,2,3,\dots,N$, generate a random parameter value

$$\lambda_n \in [0,1]$$

where λ_n represents the n^{th} firefly and N denotes the total number of fireflies in the population. Here parameter λ_n is used to convert the unit variable cost for shipping of each stage.

Step 2: For each firefly n , construct the convert unit variable cost matrices using the above Equation (6) and (7).

Step 3: For each firefly n , construct the MILP model for the TS-FCTP using with above convert unit variable cost.

Objective function

$$\text{Minimize } \sum_{i=1}^I \sum_{j=1}^J (x_{ij} b_{ij}^{\lambda_n} + f_{ij} z_{ij}) + \sum_{j=1}^J \sum_{k=1}^K (y_{jk} c_{jk}^{\lambda_n} + g_{jk} w_{jk}) \quad (8)$$

Constraints are same with TS-FCTP.

Step 4: Solve the MILP model for each firefly using the

Python. Extract the result allocation matrices.

Stage 1: $X_n = [x_{ij}]$

Stage 2: $y_n = [y_{jk}]$

Calculate the total transportation cost using Equation (5) with respective above allocation matrices for each firefly " Z_n " using the CBC in Python.

Step 5: using the obtained cost for each firefly, perform iterative updates using the modified firefly algorithm.

Calculate the brightness of each firefly using " Z_n " brightness = $1/Z_n$ for the minimization function.

Consider two fireflies i and j if $Z_i < Z_j$, then firefly i moves towards firefly j .

Compute the modified distance between two fireflies as follow

$$r_{ij} = \|\lambda_j - \lambda_i\| \quad (9)$$

Compute the modified attraction

$$\beta = \beta_0 e^{-\gamma \|\lambda_j - \lambda_i\|^2} \quad (10)$$

Update the firefly position

$$\lambda_i = \lambda_i + \beta (\lambda_j - \lambda_i) + \alpha (\text{rand} - 0.5) \quad (11)$$

$$\text{update } \lambda_i \text{ clipping} = \begin{cases} 1, & \text{if } \lambda_i > 1 \\ 0, & \text{if } \lambda_i < 0 \end{cases}$$

Repeat the MILP completing all iterations for the update λ_i

The parameters values used in the proposed algorithm are ($\alpha=0.25, \beta_0=1$ & $\gamma=1$)

Step 6: After completing all the iterations, identify the best allocation matrices with minimum total transportation cost.

The Python code for the proposed hybrid algorithm, the modified firefly model, is provide in the appendix.

RESULTS AND DISCUSSION

Numerical illustration for the proposed algorithm

To study the efficient and application of the proposed hybrid firefly algorithm, a numerical small example of TS-FCTP considered. The problem consists of two manufactures, three distribution state and four customers. The problem can be represented as follow:

Supply	(44 46)
Demand	(30 15 15 30)
Stage 1 Unit shipping cost	$\begin{pmatrix} 4 & 5 & 6 \\ 3 & 8 & 6 \end{pmatrix}$
Fixed cost	$\begin{pmatrix} 10 & 1 & 8 \\ 5 & 11 & 5 \end{pmatrix}$
Stage 2 Unit shipping cost	$\begin{pmatrix} 3 & 5 & 4 & 7 \\ 6 & 0 & 3 & 5 \\ 7 & 7 & 6 & 4 \end{pmatrix}$
Fixed cost	$\begin{pmatrix} 7 & 8 & 6 & 9 \\ 5 & 7 & 8 & 20 \\ 6 & 9 & 6 & 8 \end{pmatrix}$

Also, the table 1 illustrates the overall procedure of computational experiment was executed in Visual Studio applying the proposed algorithm solve the TS-FCTP. Code development. The algorithm was implemented in Python, and

Table 1: computational result of the proposed algorithm obtained from the python implementation

Starting Hybrid Firefly Optimization		
Iteration 1	Iteration 2	Iteration 3
Lambda: 0.417	Lambda: 0.644	Lambda: 0.644
Stage-1 Allocation X:	Stage-1 Allocation X:	Stage-1 Allocation X:
[[29 15 0]	[[0 15 29]	[[0 15 29]
[46 0 0]]	[45 0 1]]	[45 0 1]]
Stage-2 Allocation Y:	Stage-2 Allocation Y:	Stage-2 Allocation Y:
[[30 0 15 30]	[[30 0 15 0]	[[30 0 15 0]
[0 15 0 0]	[0 15 0 0]	[0 15 0 0]
[0 0 0 0]]	[0 0 0 30]]	[0 0 0 30]]
Total Cost: 734	Total Cost: 707	Total Cost: 707
Lambda: 0.72	Lambda: 0.72	Lambda: 0.527
Stage-1 Allocation X:	Stage-1 Allocation X:	Stage-1 Allocation X:
[[0 30 14]	[[0 30 14]	[[0 15 29]
[30 0 16]]	[30 0 16]]	[45 0 1]]
Stage-2 Allocation Y:	Stage-2 Allocation Y:	Stage-2 Allocation Y:
[[30 0 0 0]	[[30 0 0 0]	[[30 0 15 0]
[0 15 15 0]	[0 15 15 0]	[0 15 0 0]
[0 0 0 30]]	[0 0 0 30]]	[0 0 0 30]]
Total Cost: 724	Total Cost: 724	Total Cost: 707
Lambda: 0.0	Lambda: 0.566	Lambda: 0.566
Stage-1 Allocation X:	Stage-1 Allocation X:	Stage-1 Allocation X:
[[44 0 0]	[[0 15 29]	[[0 15 29]
[46 0 0]]	[45 0 1]]	[45 0 1]]
Stage-2 Allocation Y:	Stage-2 Allocation Y:	Stage-2 Allocation Y:
[[30 15 15 30]	[[30 0 15 0]	[[30 0 15 0]
[0 0 0 0]	[0 15 0 0]	[0 15 0 0]
[0 0 0 0]]	[0 0 0 30]]	[0 0 0 30]]
Total Cost: 794	Total Cost: 707	Total Cost: 707
--- Lambda Updates ---	--- Lambda Updates ---	--- Lambda Updates ---
Updated Lambdas: [0.644 0.72 0.566]	Updated Lambdas: [0.644 0.527 0.566]	Updated Lambdas: [0.644 0.527 0.566]
Best Cost This Iteration: 724	Best Cost This Iteration: 707	Best Cost This Iteration: 707
Best Lambda: 0.6442510123000666		
Best Cost: 707		
Execution Time: 1.2774937152862549		
Best Allocation Matrix X (Stage 1)		
[[0 15 29]		
[45 0 1]]		
Best Allocation Matrix Y (Stage 2)		
[[30 0 15 0]		
[0 15 0 0]		
[0 0 0 30]]		

Accordingly result, for the first iteration, three fireflies were initialized with random three λ values (0.417 0.720 0.000)

The total transportation cost obtained using MILP using Equation 5, based on the unit shipping cost converted according to the λ values, can be expressed as follows:

When $\lambda=0.417$, total transportation cost=734

When $\lambda=0.720$, total transportation cost=724

When $\lambda=0.000$, total transportation cost=794

The best solution in this iteration was obtained by firefly 2 with a cost of 724. The best $\lambda=0.720$

The Firefly algorithm updates the position of each

other fireflies using the attractiveness-based movement Equation 11 in iteration 2. The update rule for the parameter λ is given by

(0.644 0.720 0.566)

By repeating the above process, the minimum solution was obtained after 3 iterations with a total transportation cost 707.

Comparative analysis

A comparison analysis of the proposed algorithm for the problems related to these studies is shown in the table below.

Table 2: Comparative analysis for numerical example (Calvete *et al.*, 2018). The symbol “=” represent that the algorithm reaches the optimum solution.

Ex (i×j×k)	cplex Zopt	A. Jawahar & N. Balaji (Jawahar & Balaji, 2009) Zbest	A. Jawahar & N. Balaji (Balaji & Jawahar, 2010) Zbest	D. Raj & C. Rajendran (Antony Arokia Durai Raj & Rajendran, 2012) Zbest	D. Raj & C. Rajendran (Antony Arokia Durai Raj & Rajendran, 2012) Zbest	TSFC-M (Calvete <i>et al.</i> , 2018) Zbest	Proposed algorithm Zbest
1.(2×2×3)	112600	=	=	=	=	=	=
2.(2×2×4)	237750	=	=	=	=	=	=
3.(2×2×5)	180450	=	=	=	=	=	=
4.(2×2×6)	165650	=	=	=	=	=	=
5.(2×2×7)	162490	=	=	=	=	=	=
6.(2×3×3)	59500	=	=	=	=	=	=
7.(2×3×4)	32150	=	=	=	=	=	=
8.(2×3×6)	65945	69970	70480	67300	=	=	=
9.(2×3×8)	258730	264680	263000	=	=	=	=
10.(2×4×8)	77400	85200	80400	84600	=	=	=
11.(2×5×6)	75065	94565	95215	80865	=	=	=
12.(3×2×4)	47140	=	=	=	=	=	=
13.(3×2×5)	175350	178950	178950	178950	=	=	=
14.(3×3×4)	57100	=	=	61000	=	=	=
15.(3×3×5)	152800	=	=	156900	=	=	=
16.(3×3×6)	132890	=	=	=	=	=	=
17.(3×3×7)	99095	106615	104115	106745	=	=	=
18.(3×3×7)	281100	302350	287360	295060	=	=	=
19.(3×4×6)	76900	83500	77250	81700	=	=	=
20.(4×3×5)	118450	=	=	=	=	=	=

The small- and medium-sized problems related to TS-FCTP cases tested in this study were obtained from sites.google.com/site/fctpdataset/ (Calvete *et al.*, 2018). They are mathematical problems ranging from seven to fourteen nodes with twenty cases. The solutions for those problems were observed using the proposed algorithm. The results obtained are listed in Table 2 above, along with a comparison with the research article published by Herminia I. Calvete *et al.* in 2018 (Calvete *et al.*, 2018). In that, for those cases, the proposed hybrid algorithm was able to obtain results similar to the optimal solutions

provided by CPLEX Studio version 12.6 (Calvete *et al.*, 2018). The corresponding allocation matrices and minimum total costs are also given in the Appendix section below. This test was performed using the Python programming language on a computer with a technical specification of a “Intel(R) Core (TM) i3-1005G1 CPU @ 1.20GHz (1.20 GHz), 4.00GB RAM” capacity and provides the optimal solution in each run of the test. The average duration of the 10 iterations ranges from 0.5 to 4.5 seconds. In this research paper for small- scale problems, the CBC solver in Python (solver = pl.PULP_

CBC_CMD(msg=0, timeLimit=120, gapRel=0.02) was used to obtain the allocation matrices corresponding to each λ_n and it was used for every feasible solution within 2% range of optimal solution. According to the result allocation matrices used to obtained the total transportation cost using Equation 5. Although it easy to obtain optimal solution by CBC solver for small-scale problems, it is difficult to obtain it in a reasonable time when dealing with large- scale problems. Therefore, obtaining the necessary allocation matrices with satisfy all constraints can be done by this proposed firefly model corresponding to each λ_n . According the future expectation of this research to study the method with minimize the total transportation cost for large scale TS-FCTP.

This study was primarily conducted to research how the Firefly Algorithm could be adapted to address the transportation problem. Through this process, accurate solutions were successfully obtained for small- and medium-scale problems. Consequently, the study aimed to gain insight into how this algorithm could be applied to achieve accurate solutions for large-scale problems.

CONCLUSION

The main objective of this research paper was to study a new approach to obtain solutions for TS-FCTP by modifying the firefly model. In it, the proposed method was improved as a hybrid algorithm related to the firefly model by analysing it as a MILP problem in Python. In it, the accuracy of the algorithm was tested through several mathematical problems. In it, 20 problems ranging from seven to fourteen node were tested. In it, the results of the proposed algorithm were always similar to the optimal solution. Also, the computational time was shown to be within a reasonable range of the relevant target solution values. Also, further improving this proposed algorithm and applying it to large-scale problems can be mentioned as a future research extension. Also, the goal is to obtain more accurate solutions for the large-scale TS-FCTP in reasonable computing time.

Acknowledgements

Financial assistance from the Postgraduate institute of science, University of Peradeniya under grant number PGIS/2024/04 is acknowledge.

REFERENCES

- A. D.Raj, & Rajendran, C. (2012). A genetic algorithm for solving the fixed-charge transportation model: Two-stage problem. *Computers & Operations Research*, 39(9), 2016–2032. <https://doi.org/10.1016/j.cor.2011.09.020>
- Balaji, A. N., & Jawahar, N. (2010). A simulated annealing algorithm for a two-stage fixed charge distribution problem of a supply chain. *International Journal of Operational Research*, 7(2), 192–215. <https://doi.org/10.1504/IJOR.2010.030803>
- Calvete, H. I., Galé, C., & Iranzo, J. A. (2016). An

- improved evolutionary algorithm for the two-stage transportation problem with fixed charge at depots. *OR Spectrum*, 38(1), 189–206. <https://doi.org/10.1007/s00291-015-0416-9>
- Calvete, H. I., Galé, C., Iranzo, J. A., & Toth, P. (2018). A matheuristic for the two-stage fixed-charge transportation problem. *Computers & Operations Research*, 95, 113–122. <https://doi.org/10.1016/j.cor.2018.03.007>
- Cosma, O., Dănculescu, D., & Pop, P. C. (2019). On the two-stage transportation problem with fixed charge for opening the distribution centers. *IEEE Access*, 7, 113684–113698. <https://doi.org/10.1109/ACCESS.2019.2936095>
- Cosma, O., Pop, P. C., & Dănculescu, D. (2020). A novel matheuristic approach for a two-stage transportation problem with fixed costs associated to the routes. *Computers & Operations Research*, 118, Article 104906. <https://doi.org/10.1016/j.cor.2020.104906>
- Gen, M., Altıparmak, F., & Lin, L. (2006). A genetic algorithm for two-stage transportation problem using priority-based encoding. *OR Spectrum*, 28(3), 337–354. <https://doi.org/10.1007/s00291-005-0029-9>
- Jawahar, N., & Balaji, A. N. (2009). A genetic algorithm for the two-stage supply chain distribution problem associated with a fixed charge. *European Journal of Operational Research*, 194(2), 496–537. <https://doi.org/10.1016/j.ejor.2007.12.005>
- Jayakumar Arumugham, A., Krishnaraj, C., & Raghunayagan, P. (2017). Optimization of fixed charge problem in Python using PuLP package. *International Journal of Control Theory and Applications*, 10(2), 443–447.
- Jha, A. K., Sharma, G. S., & Kaushik, N. (2023). Modeling of transport problem in linear programming with Python (PULP). *samriddhi: A Journal of Physical Sciences, Engineering and Technology*, 15(1), 104–108. <https://doi.org/10.18090/samriddhi.v15i01.14>
- Mitchell, S. (2011). *PuLP: A linear programming toolkit for Python* [Computer software].
- Pop, P. C., Sabo, C., Biesinger, B., Hu, B., & Raidl, G. R. (2017). Solving the two-stage fixed-charge transportation problem with a hybrid genetic algorithm. *Carpathian Journal of Mathematics*, 33(3), 365–371. <https://doi.org/10.37193/CJM.2017.03.11>
- Thakur, S. K. (2025). *A comprehensive model for two-stage fixed-charge transportation planning problem*. Zenodo. <https://doi.org/10.5281/zenodo.15304585>
- Yang, X.-S. (2010). *Nature-inspired metaheuristic algorithms*. Luniver Press.

Appendix

```
# HYBRID FIREFLY
```

```
# =====
```

```
def hybrid_firefly(data,
                    n_fireflies=3,
                    iterations=10,
```

```

alpha=0.25,
beta0=1,
gamma=1):
lambdas = np.random.rand(n_fireflies)
pool = mp.Pool(mp.cpu_count())
for it in range(iterations):
    osts = pool.map(
        evaluate_firefly,
        [(data, lam) for lam in lambdas]
    )
    costs = np.array(costs)
    for i in range(n_fireflies):
        for j in range(n_fireflies):
            if costs[j] < costs[i]:
                rij = abs(lambdas[i] - lambdas[j])

```

```

beta = beta0 * np.exp(-gamma * rij**2)
lambdas[i] = (
    lambdas[i]
    + beta * (lambdas[j] - lambdas[i])
    + alpha * (np.random.rand() - 0.5)
)
lambdas[i] = np.clip(lambdas[i], 0,
    print(f"Iteration {it+1} | Best Cost: {costs.min()}")
pool.close()
pool.join()
best_index = np.argmin(costs)
best_lambda = lambdas[best_index]
best_X, best_Y, best_cost = solve_two_stage_fctp(data,
    best_lambda)
return best_lambda, best_X, best_Y, best_cost

```

1.	2.
<pre> ===== Best Lambda: 0.695067374119903 Best Cost: 112600.0 Execution Time: 1.5334997177124023 ===== Best Allocation Matrix X (Stage 1): [[450 0] [550 0]] Best Allocation Matrix Y (Stage 2): [[85 650 265] [0 0 0]] </pre>	<pre> ===== Best Lambda: 0.8582889248641495 Best Cost: 237750.0 Execution Time: 1.2616257667541504 ===== Best Allocation Matrix X (Stage 1): [[1000 0] [1600 0]] Best Allocation Matrix Y (Stage 2): [[600 450 750 800] [0 0 0 0]] </pre>
3.	4.
<pre> ===== Best Lambda: 0.6434394195529682 Best Cost: 188450.0 Execution Time: 1.5408588512420654 ===== Best Allocation Matrix X (Stage 1): [[800 0] [800 600]] Best Allocation Matrix Y (Stage 2): [[0 0 600 500 500] [350 250 0 0 0]] </pre>	<pre> ===== Best Lambda: 0.8348647943041185 Best Cost: 165650.0 Execution Time: 1.4561762809753418 ===== Best Allocation Matrix X (Stage 1): [[1050 450] [0 1800]] Best Allocation Matrix Y (Stage 2): [[600 450 0 0 0 0] [0 0 550 850 550 300]] </pre>
5.	6.
<pre> ===== Best Lambda: 0.5703088416494222 Best Cost: 162490.0 Execution Time: 0.5909740924835205 ===== Best Allocation Matrix X (Stage 1): [[1600 250] [0 2000]] Best Allocation Matrix Y (Stage 2): [[450 0 0 750 140 50 210] [0 850 1400 0 0 0 0]] </pre>	<pre> ===== Best Lambda: 0.3108179397028964 Best Cost: 59500.0 Execution Time: 0.37769103050231934 ===== Best Allocation Matrix X (Stage 1): [[0 600 0] [350 100 0]] Best Allocation Matrix Y (Stage 2): [[0 0 350] [400 300 0] [0 0 0]] </pre>
7.	8.
<pre> ===== Best Lambda: 0.46305838508685093 Best Cost: 32150.0 Execution Time: 2.7292566299438477 ===== Best Allocation Matrix X (Stage 1): [[0 250 0] [330 20 0]] Best Allocation Matrix Y (Stage 2): [[150 80 100 0] [0 0 0 270] [0 0 0 0]] </pre>	<pre> ===== Best Lambda: 0.5249636738789742 Best Cost: 65945.0 Execution Time: 1.7988994121551514 ===== Best Allocation Matrix X (Stage 1): [[0 0 525] [1175 0 0]] Best Allocation Matrix Y (Stage 2): [[0 245 175 0 555 200] [0 0 0 0 0 0] [50 0 0 475 0 0]] </pre>

9 <pre> ===== Best Lambda: 0.45451940080976894 Best Cost: 258730.0 Execution Time: 3.180098295211792 ===== Best Allocation Matrix X (Stage 1): [[1650 0 1200] [0 2280 0]] Best Allocation Matrix Y (Stage 2): [[750 0 0 0 550 0 0 350] [0 600 980 700 0 0 0 0] [0 0 0 0 0 400 800 0]] </pre>	10 <pre> ===== Best Lambda: 0.5328303301638123 Best Cost: 77400.0 Execution Time: 3.9709551334381104 ===== Best Allocation Matrix X (Stage 1): [[800 0 0 0] [0 1050 0 150]] Best Allocation Matrix Y (Stage 2): [[50 0 0 450 0 0 300 0] [100 300 0 0 250 0 0 400] [0 0 0 0 0 0 0 0] [0 0 100 0 0 50 0 0]] </pre>
11 <pre> ===== Best Lambda: 0.29908291120753633 Best Cost: 75065.0 Execution Time: 2.3457469940185547 ===== Best Allocation Matrix X (Stage 1): [[0 0 0 0 650] [1250 0 0 0 0]] Best Allocation Matrix Y (Stage 2): [[650 245 75 125 0 155] [0 0 0 0 0 0] [0 0 0 0 0 0] [0 0 0 0 0 0] [0 0 400 0 250 0]] </pre>	12 <pre> ===== Best Lambda: 0.5739189123031762 Best Cost: 47140.0 Execution Time: 1.8030543327331543 ===== Best Allocation Matrix X (Stage 1): [[250 0] [150 0] [0 300]] Best Allocation Matrix Y (Stage 2): [[140 50 0 210] [0 0 300 0]] </pre>
13 <pre> ===== Best Lambda: 0.5991881427831851 Best Cost: 175350.0 Execution Time: 1.8355472087860107 ===== Best Allocation Matrix X (Stage 1): [[1200 0] [0 800] [0 1000]] Best Allocation Matrix Y (Stage 2): [[600 450 0 0 150] [0 0 550 850 400]] </pre>	14 <pre> ===== Best Lambda: 0.40523244855678164 Best Cost: 57100.0 Execution Time: 1.5460827350616455 ===== Best Allocation Matrix X (Stage 1): [[500 0 0] [0 0 400] [100 0 0]] Best Allocation Matrix Y (Stage 2): [[250 0 0 350] [0 0 0 0] [0 350 50 0]] </pre>
15 <pre> ===== Best Lambda: 0.506578328994613 Best Cost: 152800.0 Execution Time: 1.77451753616333 ===== Best Allocation Matrix X (Stage 1): [[90 0 200] [0 0 400] [0 150 0]] Best Allocation Matrix Y (Stage 2): [[0 90 0 0 0] [0 0 0 0 150] [150 0 210 300 0]] </pre>	16 <pre> ===== Best Lambda: 0.46387938908773474 Best Cost: 132890.0 Execution Time: 1.4578619003295898 ===== Best Allocation Matrix X (Stage 1): [[910 0 0] [500 0 0] [500 0 0]] Best Allocation Matrix Y (Stage 2): [[800 300 90 120 200 400] [0 0 0 0 0 0] [0 0 0 0 0 0]] </pre>
17 <pre> ===== Best Lambda: 0.42824009377799455 Best Cost: 99095.0 Execution Time: 4.3993589878082275 ===== Best Allocation Matrix X (Stage 1): [[0 0 375] [0 1300 0] [700 0 0]] Best Allocation Matrix Y (Stage 2): [[145 0 0 0 555 0 0] [55 0 150 475 0 560 60] [0 245 0 0 0 0 130]] </pre>	18 <pre> ===== Best Lambda: 0.2861168517191677 Best Cost: 281100.0 Execution Time: 1.793264389038086 ===== Best Allocation Matrix X (Stage 1): [[800 0 0] [0 1200 0] [930 0 0]] Best Allocation Matrix Y (Stage 2): [[350 700 0 0 0 680 0] [0 0 150 240 330 0 540] [0 0 0 0 0 0 0]] </pre>
19 <pre> ===== Best Lambda: 0.6192813752277592 Best Cost: 76900.0 Execution Time: 1.8213942985534668 ===== Best Allocation Matrix X (Stage 1): [[600 0 0 600] [0 550 0 0] [0 750 0 0]] Best Allocation Matrix Y (Stage 2): [[0 0 300 0 300 0] [0 0 350 250 0 700] [0 0 0 0 0 0] [450 150 0 0 0 0]] </pre>	20 <pre> ===== Best Lambda: 0.5633223729201469 Best Cost: 118450.0 Execution Time: 2.09780216217041 ===== Best Allocation Matrix X (Stage 1): [[0 400 0] [650 0 0] [950 0 0] [0 500 0]] Best Allocation Matrix Y (Stage 2): [[450 0 650 500 0] [0 750 0 0 150] [0 0 0 0 0]] </pre>