Available online at www.garmian.edu.krd

Passer Journal of Basic and Applied Sciences

Passer 8 (Issue 2) (2026) 1047-1056

Original Article

Peregrine Falcon Stoop Optimization for Solving the Multi-Objective Travelling Salesman Problem

Ali Abbas Numman * [1]

[1] *Department of Mathematics, College of Science, Baghdad University, Baghdad 10001, Iraq.*

Received 22 January 2026; revised 12 July 2026;
accepted 23 July 2026; available online 21 September 2026

DOI: 10.24271/PSR.2026.572081.2547

ABSTRACT

The Multi-Objective Traveling Salesman Problem (MOTSP) minimizes the distance and cost simultaneously. This phenomenon leads to computational expensive status of exact algorithms, making them unsuitable for large-scale problems. This study proposes a new meta heuristic algorithm, inspired by nature, to solve the MOTSP. The proposed method, named the Peregrine Falcon Swooping Optimization (PFSO), takes any solution found using approximations and improves it by simulating the hunting and swooping actions of a peregrine falcon. In contrast to many meta-heuristic existing methods, the PFSO method does not require a search history to be stored for optimizing a solution. Through a comparative study conducted by analyzing the different methods of branching and cutting algorithms as well as local search algorithms like ant colony optimization, the results indicate that the proposed method is able to provide better results in terms of distance and cost.

Keywords: Multi-objective traveling salesman problem, peregrine falcon dive optimization, inferential algorithms, branching and cutting method, local search, ant colony optimization.

Introduction

Historically, combinatorial optimization has been depending heavily on the problem of travelling salesman regarding the solution of single-route efficiency. Though, strict condition to the minimization of distance is barely sufficient for actual applications. To reduce this gap, the Multi-Objective TSP (MOTSP) is called for allowing for the simultaneous optimization of comparing criteria, particularly distance cost and time. For example, you may see [1,2,3,4]. These aspects are considered critical maturity in routing algorithms, balacing from theoretical asspects to solvers capacity of handling multi-faceted operational objectives. It is commonly used in transportation and logistics to optimize routes for distance, time, and cost. In supply chain management, it improves delivery efficiency and resource utilization. It is also applied in manufacturing scheduling and network design to enhance performance and reduce operational costs. Furthermore, MOTSP contributes to robotics and smart city systems, enabling efficient and intelligent route planning [5,6,7].

In the Literature Survey, there are many applications of MOTSP. In 2021, the Multi-Objective Traveling Salesperson Problem with Time Windows, which models the monitoring of pilgrims during the Hajj through a multiple-salesperson MOTSP incorporating time-windows, workforce size, waiting time, and tour length. In 2022, a Hybridization of evolutionary algorithm and deep reinforcement learning for multi-objective orienteering optimization, where they decomposed the problem into a knapsack and a TSP component and solved with MOEA and DRL, respectively, to capture route-selection and sequencing simultaneously, was presented. In 2023, a two-stage evolutionary algorithm (TSEA) was developed for the bi-objective TSP that uses local search + NSGA-II in stage one, then seeded hybrid local search in stage two to improve convergence and diversity of the Pareto front. Zhao et al. In 2024 introduced the deep reinforcement learning algorithm framework for solving the multi-objective traveling salesman problem based on feature transformation, where a feature-transformed DRL model is used to learn routing under multiple conflicting objectives and demonstrate improved PF quality over conventional heuristics. Gao et al. proposed “Multi-Objective Optimization for Traveling Salesman Problem. In addition, a Multi-Objective Pointer Network (MOPN) for MOTSP that leverages transfer learning to scale from small to large instances and achieves superior performance in hypervolume and spread metrics was introduced [8,9,10,11].

* Corresponding author
E-mail address ali.abbas2303@sc.uobaghdad.edu.iq (Instructor).
Peer-reviewed under the responsibility of the University of Garmian.

Mathematical Model Of The Multi-Objective Travelling Salesman Problem (MOTSP)

Objective Function

The MOTSP seeks a vector minimization over $k \geq 2$ objectives:

$$\min_{x} F(x) = [f_1(x), f_2(x), \dots, f_k(x)],$$

where each objective has the generic linear form

$$f_m(x) = \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} w_{ij}^{(m)} x_{ij}, \qquad m = 1, \dots, k.$$

where, $w_{ij}^{(m)}$ is the weight (distance, time, cost, risk, energy) of (i, j), under objective m. The solution concept is the Pareto set of non-dominated tours.

MOTSP Requirements

1. Each city is visited exactly once, and the tour returns to the start (Hamiltonian cycle).
2. No subtours (disconnected cycles) are allowed.
3. Decision variables are binary on arcs.
4. The model must support multiple objectives and allow recovery of Pareto-optimal solutions (e.g., via scalarization or evolutionary search).

Variable Definitions

- n: number of cities; index set $V = \{1, \dots, n\}$.
- $x_{ij} \in \{0,1\}$for $i \neq j$: equals 1 if the tour travels directly from the city in to city j, 0 otherwise.
- $w_{ij}^{(m)} \geq 0$: weight of arc (i, j)in objective m, $m = 1, \dots, k$.
- $u_i \in \mathbb{Z}$(MTZ auxiliary variables) for $i = 2, \dots, n$to eliminate subtours.

Model Construction Objectives

$$\min F(x) = [f_1(x), \dots, f_k(x)], f_m(x) = \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} w_{ij}^{(m)} x_{ij} \quad (m = 1, \dots, k).$$

degree (in/out) constraints: $\sum_{j=1, j\neq i}^{n} x_{ij} = 1 \forall i \in V$, $\sum_{i=1, i\neq j}^{n} x_{ij} = 1 \; \forall j \in V.$

Subtour-elimination (MTZ) constraints:

$$u_i - u_j + n\, x_{ij} \leq n - 1 \forall i \neq j,\ i, j \in \{2, \dots, n\}, \qquad 2 \leq u_i \leq n \forall i \in \{2, \dots, n\}.$$

$$x_{ij} \in \{0,1\}(i \neq j), u_i \in \mathbb{Z}.$$

Solving Methods for MOTSP

Exact Methods

The exact solution methods for the Multi-Objective Travelling Salesman Problem (MOTSP) aim to find globally optimal Pareto solutions with mathematical precision. The most prominent exact approaches include Dynamic Programming, Integer Linear Programming (ILP), Branch and Bound, and Branch and Cut. Among these, the Branch and Cut method stands out for its efficiency and speed. Taking cutting-plane constraints immediately to the Branch and Bound structure, the procedure directly edge-cuts the search tree, ignoring unbeneficial regions at early stages of the process. This algorithm allows the solver to obtain exact solutions without an extensive node checking, initially starting it as the standard approach for small scale to large MOTSP where computing efficiency should not be compromised. With respect to solution accuracy.

Heuristics Methods

Heuristic approximation method has been effectively used in combinatorial optimization. Its algorithm gives a useful classification. Therefore, a functional distinction is often taken between two essential branches of thought:

1. Constructive Heuristic: Its algorithms work on an accumulative base. Beginning from an initial case (an empty set), it assembles a solution step by step, strictly attaching the most active element at each stage until a complete, structure is reached.

2. Meta-Heuristic: Oppositely, this function plays a role as main structure. Instead of starting initially, it orchestrates the solution process. It derives subordinate algorithms to avoid local optima, providing a problem-agnostic strategy that seeks global optimization regardless of the specific constraints. For example there are several efficient methods that have been studied include but not limited to; Simulated Annealing (SA), Tabu Search (TS), Stochastic Local Search (SLS), Particle Swarm Optimization (ACO), and the Genetic Algorithm (GA) [12,13,14].

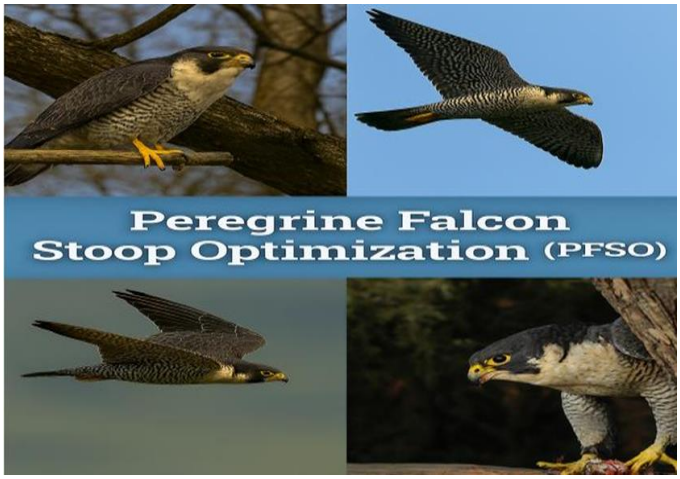
Genetic Algorithm (GA) is an optimization method inspired by natural evolution. It works by evolving a population of candidate solutions through selection, crossover, and mutation. Fitter solutions have a higher chance of producing improved offspring in each generation. GA is widely used to solve complex optimization problems where exact methods are too slow [14]. Practical Swarm Optimization (ACO) is a population-based algorithm inspired by the collective movement of birds and fish. Each particle represents a candidate solution that moves through the search space by learning from its own best position and the swarm's best. Particles adjust their velocity to balance exploration and exploitation. ACO is simple, fast, and effective for continuous optimization problems [15,16,17].

New Techniques For Solving MOTSP

In this section, we introduce new methods for solving MOTSP.

Peregrine Falcon Stoop Optimization (PFSO)

A novel algorithm was proposed for solving the Multi-Objective Traveling Salesman Problem (MOTSP), Most of the meta-heuristic methods that have been introduced so far is memory-based method such as (GA, ASO, ACO, TS) here we introduce new meta-heuristic that employee special relations and techniques that directly aim to ensuring to improve the solutions The Peregrine Falcon, known and admired for its impressive aerial maneuvers, employs a variety of strategies to secure a satisfying meal, often relying on fast pursuits, rapid dives, and other skilled tactics [18,19]. The Stoop, widely recognized as one of its most renowned hunting methods, is employed by the Peregrine Falcon. Utilizing its exceptional vision, the falcon soars to great heights, scanning the skies for pray beneath it. "Upon locking onto prey, the falcon retracts its wings to minimize aerodynamic drag, initiating a high-velocity dive technically termed a 'stoop.' In this state, the bird effectively becomes a projectile, capable of achieving terminal velocities that surpass 322 kph (200 mph) [18]. With precision and agility, the falcon clenches its talons and employs them as a precise weapon, striking the prey with the accuracy of a guided missile to eliminate the pray first then another stoop to snatching it.



Feature Falcon Stoop system for MOTSP

As we attempt to put tsp to simulation to the Peregrine Falcon hunt strategy, we must list it's hunting technique:

1. Scanning: Using its extraordinary vision to scan for prey beneath it.
2. choosing suitable pray as target.
3. Stoop: The Peregrine Falcon is dive-bombing back down at breath-taking speeds two times as follows:

i. Striking the prey with high accuracy.
ii. Another stoop to snatch up the pray.

The key parameters required for the successful implementation of the Peregrine Falcon Stoop Optimization (PFSO) to the MOTSP are succinctly summarized in Figure (1).

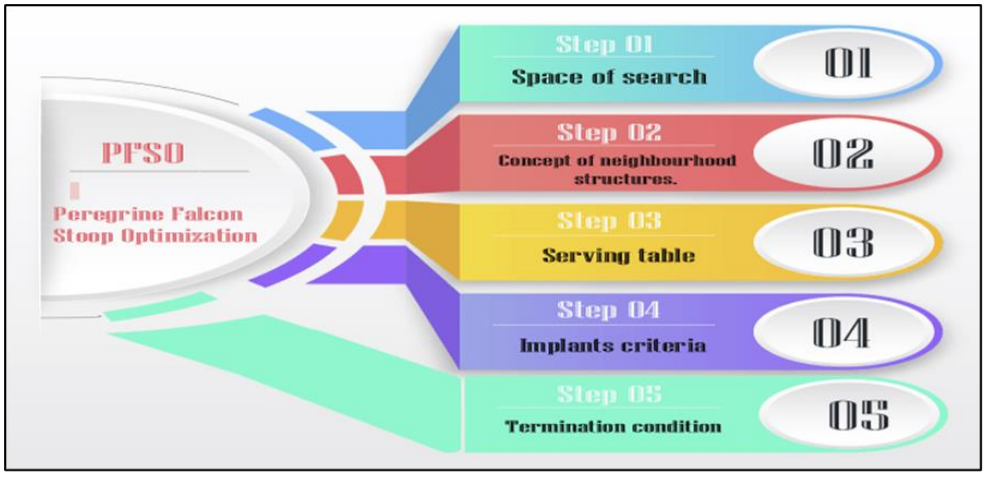
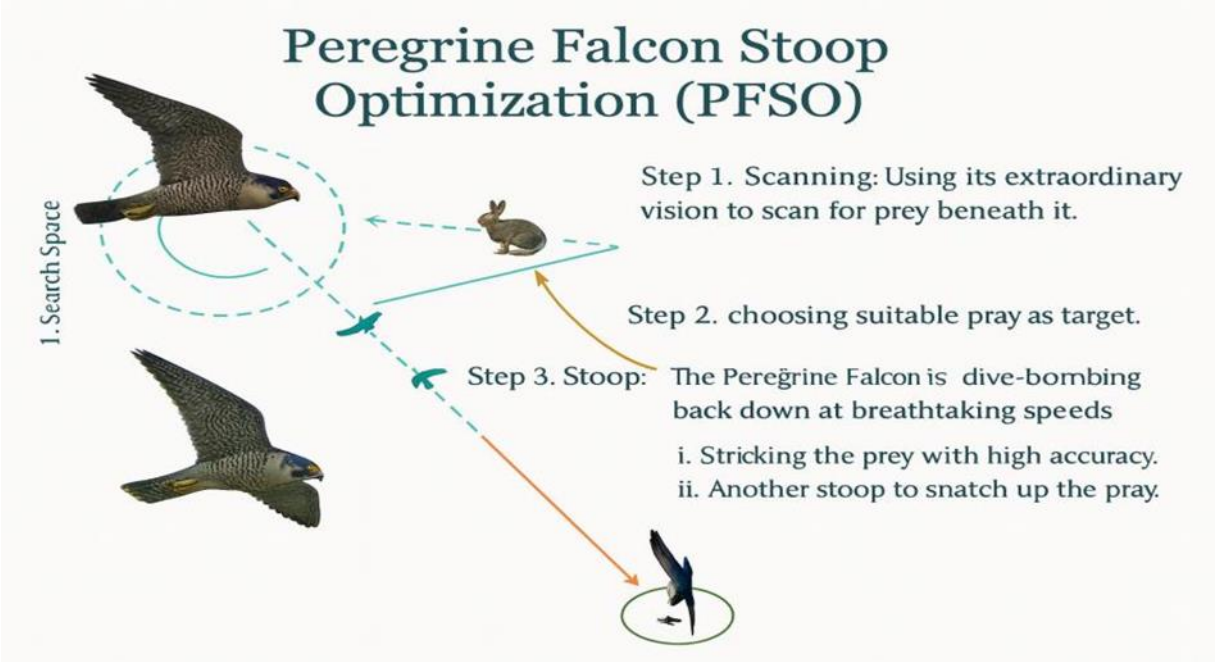


Figure 1: key parameters for PFSO.



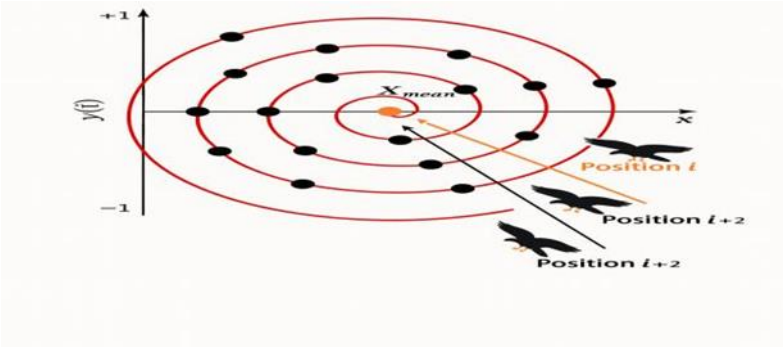
Space of search.

The search space can be considered as first step of The Peregrine Falcon hunting technique, here we begin our progress by constructions of table contains the cities pairs with their distance and cost then sort them acceding. Then start from top of the table and take first pair that is not satisfied as put together on route as first iteration.

Concept of neighborhood structures.

The creation of the neighborhood structure involves implementing a "move" on the existing solution during each iteration within the search space by applying come special relations design for this purpose to ensure high accuracy.

The neighborhood structure can be put in simulation of The Peregrine Falcon hunting techniques as it's strike pray as in forcing customer into specific locations and as is snatching some other customer from that route (by using special relation) if it's exceeding the MOTSP.



Serving table.

Serving table is the locations of all cities that taking away from it's original route by The Peregrine Falcon snatching.

Implants criteria.

Implants criteria is special technique that allows us to put all the customer on specific route to ensure all the customers are served.

Termination condition.

At some point, the search process in a Peregrine Falcon Stoop Optimization must reach its conclusion. To terminate the optimization, there are various potential approaches that can be employed, such as:

1. After reaching a set number of iterations.

2. After a specific number of iterations have elapsed without any improvement in the objective function's value,

3. A certain condition is met.

Algorithm: Peregrine Falcon Stoop Optimization (PFSO)

Input

1. Number of cities n.
2. Distance matrix $D = [d_{ij}]_{n\times n}$.
3. Cost matrix $C = [c_{ij}]_{n\times n}$(possibly $C = D$).

Output

- Best tour $T^{\backslash *}$.
- Best cost $C(T^{\backslash *})$.

Phase A – Initial Solution (Approximation / Cheapest Insertion)

Step A1. Read input data

$$n,\ d_{ij},\ c_{ij}, i, j = 1, \dots, n.$$

Choose a start city $s \in V$(e.g., $s = 1$).

Step A2. Construct an initial feasible solution $T^{(0)}$using an approximation method

1. Initialize:

$$T = [s], R = V \setminus \{s\}.$$

Insert the nearest neighbor:

$$j^{\backslash *} = \arg\min_{j\in R} c_{sj}, T \leftarrow [s, j^{\backslash *}], R \leftarrow R \setminus \{j^{\backslash *}\}.$$

2. While $R \neq \emptyset$:
   - For each city $u \in R$
   - For each edge (T_k, T_{k+1})in the current tour
   - Compute insertion cost increase:

$$\Delta(u, k) = c_{T_k u} + c_{u T_{k+1}} - c_{T_k T_{k+1}}.$$

   - Choose global minimum:

$$(u^{\backslash *}, k^{\backslash *}) = \arg\min_{u\in R,\ k} \Delta(u, k).$$

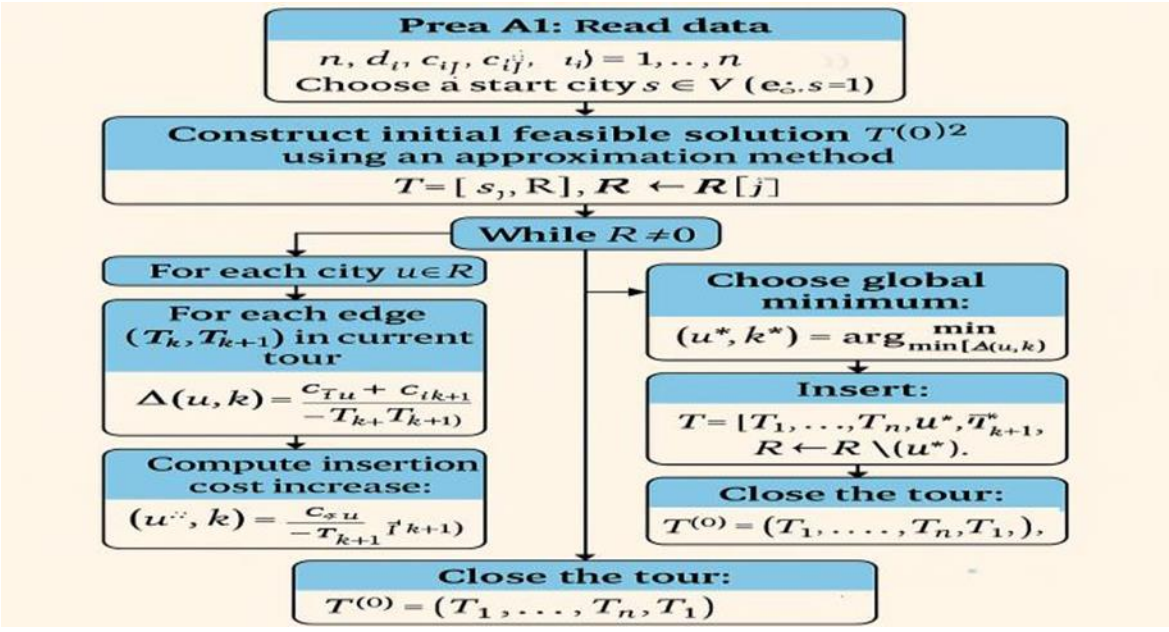
   - Insert:

$$T \leftarrow [T_1, \dots, T_{k^{\backslash *}}, u^{\backslash *}, T_{k^{\backslash *}+1}, \dots], R \leftarrow R \setminus \{u^{\backslash *}\}.$$

3. Close the tour:

$$T^{(0)} = (T_1, \dots, T_n, T_1).$$

So, after Phase A we have an initial tour $T^{(0)}$and its cost $C(T^{(0)})$.



Phase B – Pair Space Construction (Stooping Targets)

This corresponds to your Step 3.

Step B1. Construct the full pair list:

$$\mathcal{P} = \{(i, j) \mid 1 \le i < j \le n\}.$$

Number of pairs:

$$|\mathcal{P}| = r = \binom{n}{2} = \frac{n^2 - n}{2}.$$

Step B2. For each pair $(i, j) \in \mathcal{P}$, compute a scalar priority, for example:

- distance-based:

$$\pi(i,j) = d_{ij},$$

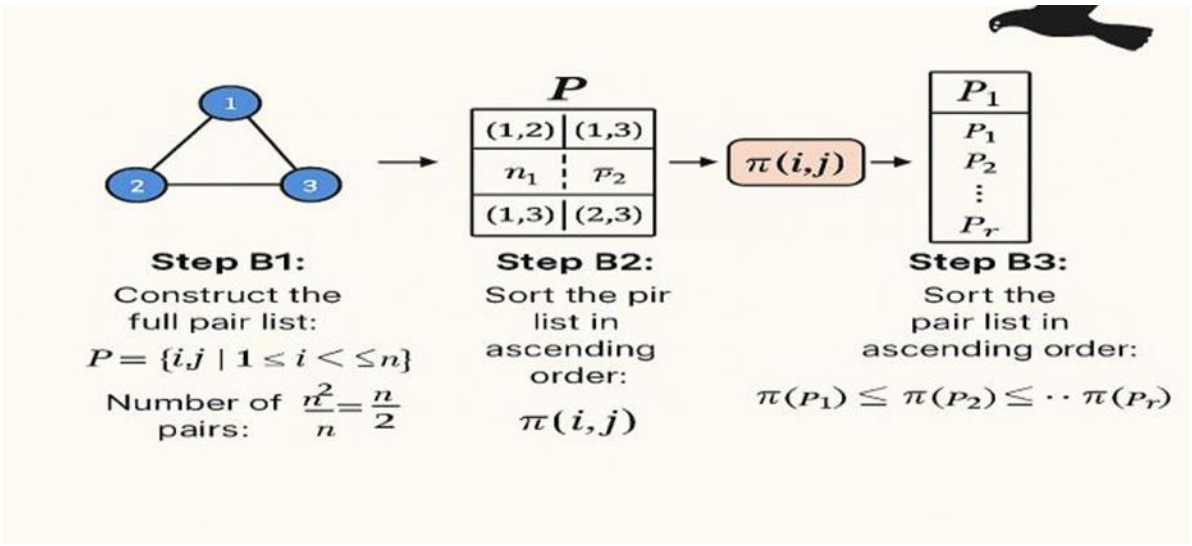
- or cost-based:

$$\pi(i,j) = c_{ij}.$$

Step B3. Sort the pair list in ascending order:

$$(P_1, P_2, \dots, P_r) \text{such that} \pi(P_1) \le \pi(P_2) \le \cdots \le \pi(P_r).$$

This defines the search space of "stoop targets" (like the falcon choosing which prey pair to attack).



Phase C – PFSO Movement (K–pair Falcon Stoop)

Now we define the movement equations M_1, $M_2$and the general M(i).

C.1 – One pair movement (basic stoop)

Take a pair $P = (i,j)$from the sorted list. Assume in the current tour T, city jhas two neighbors:

- left neighbor: $\beta_L(j)$,
- right neighbor: $\beta_R(j)$.

Similarly, city ihas neighbors $\beta_L(i)$, $\beta_R(i)$.

You defined two movement options:

(1) Move $\mathbf{P_i}$next to $\mathbf{P_j}$

You consider two possible sides of j:

- Place ion the right of j: $\beta_1 = \beta_R(j)$.
- Place ion the left of j: $\beta_2 = \beta_L(j)$.

For each side $t \in \{1,2\}$, define:

$$M_1(t) = D(P_i, P_j) + D(P_i, \beta_t) - D(P_j, \beta_t).$$

More explicitly:

- For right side:

$$M_1(1) = d_{ij} + d_{i\,\beta_R(j)} - d_{j\,\beta_R(j)}.$$

For left side:

$$M_1(2) = d_{ij} + d_{i\,\beta_L(j)} - d_{j\,\beta_L(j)}.$$

This is the movement cost to "attach" city iaround city j.

(2) Move $\mathbf{P_j}$next to $\mathbf{P_i}$

Similarly, two options:

- $\beta_1 = \beta_R(i)$(right of i),
- $\beta_2 = \beta_L(i)$(left of i).

For each $t \in \{1,2\}$:

$$M_2(t) = D(P_i, P_j) + D(P_j, \beta_t) - D(P_i, \beta_t).$$

Explicitly:

- Right of i:

$$M_2(1) = d_{ij} + d_{j\,\beta_R(i)} - d_{i\,\beta_R(i)}.$$

Left of i:

$$M_2(2) = d_{ij} + d_{j\,\beta_L(i)} - d_{i\,\beta_L(i)}.$$

So, for one pair, we have 4 candidate moves:

$$\{M_1(1), M_1(2), M_2(1), M_2(2)\}.$$

We choose the minimal:

$$M(P) = \min\{M_1(1), M_1(2), M_2(1), M_2(2)\}.$$

This is the stoop cost for pair P.

C.2 – K-pair generalization

When you consider all pairs, the number of movement equations (candidates) is roughly:

$$M(i) = n(n-1)$$

as you wrote, because for ncities we can generate on the order of $n(n-1)$pair-moves.

For K-pair PFSO in one outer iteration:

1. From the sorted pair list $(P_1, \dots, P_r)$, select the first Keligible pairs that are:
   - not already adjacent in the tour,
   - not conflicting with each other (no overlapping structural constraints, if you impose that).

Let this selected subset be:

$$\mathcal{P}_K = \{P^{(1)}, P^{(2)}, \dots, P^{(K)}\}.$$

2. For each selected pair $P^{(k)}$, compute its stoop movement cost:

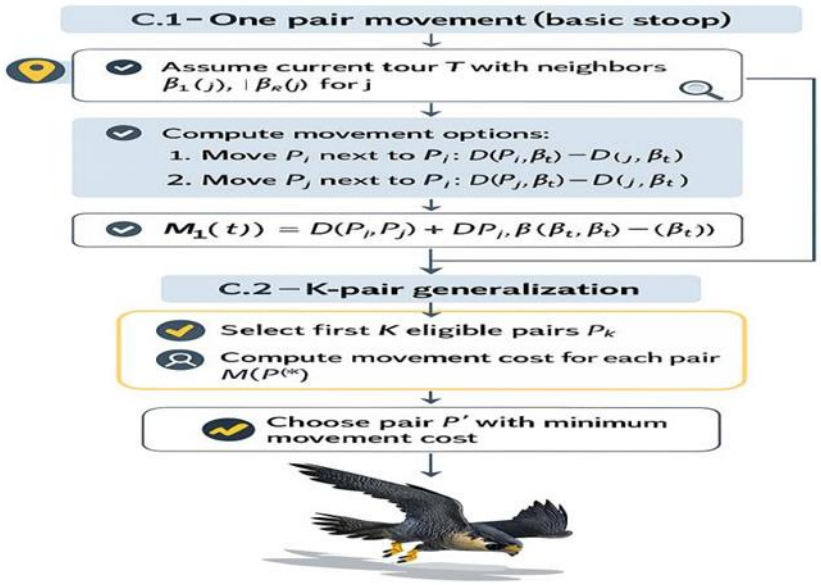
$$M(P^{(k)}) = \min\{M_1^{(k)}(1), M_1^{(k)}(2), M_2^{(k)}(1), M_2^{(k)}(2)\}.$$

3. Among all candidate pair moves (or among all K-pairs if you treat them jointly), choose the move with global minimum movement cost:

$$P^{\backslash *} = \arg\min_{P \in \mathcal{P}_K} M(P).$$

4. Update the tour Tby applying the movement associated with $P^{\backslash *}$(move the corresponding city to its new position in the route).

This is the falcon "stooping" on the best prey (pair) according to the movement function M.



Phase D – Local Refinement (2-opt Best-Improvement, Special Case K=1)

Now we connect this to your MATLAB code.

In your code, after constructing the initial tour, you use 2-opt (Best-Improvement) until convergence:

For a tour $T = (v_1, \dots, v_n, v_1)$, a 2-opt move is defined by indices $2 \le i < k \le n-1$.
The move reverses the segment $(v_i, \dots, v_k)$.

The cost change is:

$$\Delta(i,k) = (c_{v_{i-1}v_k} + c_{v_i v_{k+1}}) - (c_{v_{i-1}v_i} + c_{v_k v_{k+1}}).$$

A move is accepted if:

$$\Delta(i,k) < 0.$$

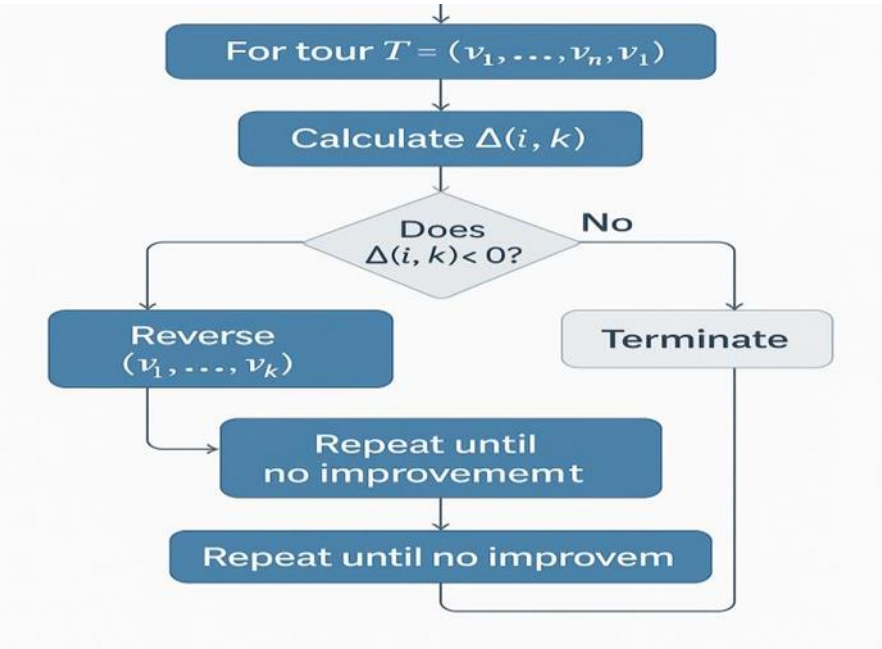
Best-Improvement strategy:

- Evaluate $\Delta(i,k)$for all valid (i,k).
- Find:

$$(i^{\backslash *}, k^{\backslash *}) = \arg\min_{i,k} \Delta(i,k) \text{subject to} \Delta(i^{\backslash *}, k^{\backslash *}) < 0.$$

- Apply the best move, update T.

Repeat until no improving move exists.



Applied The Proposed Techniques to Solve MOTSP

To evaluate the proposed techniques, five random test instances were generated for each problem size $n = 8$ to $n = 300$. The city coordinates (X, Y) were uniformly distributed within the range $[-10,10]$, and the travel costs were randomly assigned within $[1,10]$. The newly developed PFSO method was then applied and compared against ACO algorithm, with all results benchmarked against the optimal solutions obtained using the Branch and Cut (BAC) approach.

Note: For all tables, we give the following notations:

OP: Optimal value of the objective function.

BV: Best value of objective functions.

T: CPU-time in seconds.

POP: Percentage of BV for OP, s.t.

$\text{POP} = \frac{\text{BV}}{\text{OP}} \times 100\%$.

R: Time less than 1 second, $R \in (0,1)$.

TM: Total mean.

Best: Best value among heuristic methods when the optimal value is inefficient.

Table 1 shows a comparison of the results of the sets of simulation random examples between BAC, PFSO and ACO for $n = 8$ to $n = 175$.

Table 2 shows a comparison of the results of the sets of simulation random examples between PFSO and ACO for ($n = 200$ to $n = 300$) where BAC becomes inefficient.

Table 1: Comparing results of BAC, ACO and PFSO for (n=8 to n=175).

n	BAC		ACO		PFSO (K = 1)		PFSO (K = 3)		POP		
	OP	Time	BV	T	BV	T	BV	T	ACO	PFSO (K = 1)	PFSO (K = 3)
8	50.99	R	50.99	R	50.99	R	50.99	R	100%	100%	100%
9	69.15	R	69.15	R	70.42	R	69.15	R	100%	98%	100%
10	69.15	R	69.99	R	70.42	R	69.15	R	99%	98%	100%
12	70.69	R	74.35	R	73.53	R	70.69	R	95%	96%	100%
14	74.81	R	74.81	R	76.05	R	74.81	R	100%	98%	100%
16	66.59	R	66.59	R	67.39	R	66.59	R	100%	99%	100%
18	79.91	R	79.91	R	80.59	R	79.91	R	100%	99%	100%
20	76.99	R	78.35	R	83.28	R	76.99	R	98%	92%	100%
30	95.54	R	103.33	R	103.62	R	96.36	R	92%	92%	99%

40	107.13	R	112.51	R	110.87	R	110.79	R	95%	97%	97%
50	120.18	R	128.30	2.41	127.15	R	124.54	R	94%	95%	96%
60	126.78	1.62	130.22	3.73	143.93	R	130.22	1.22	97%	88%	97%
70	134.50	1.69	141.79	5.58	147.93	R	137.17	1.63	95%	91%	98%
80	143.93	4.87	156.30	8.23	151.79	R	148.75	1.70	92%	95%	97%
90	148.93	19.38	157.63	10.70	165.87	R	149.02	6.31	94%	90%	100%
100	159.64	2.44	169.86	15.01	173.77	R	166.19	6.25	94%	92%	96%
125	176.00	83.31	186.84	26.49	198.20	R	182.81	18.83	94%	89%	96%
150	201.52	58.20	220.89	43.42	227.66	R	217.58	72.12	91%	89%	93%
175	208.23	991.89	232.11	66.61	229.86	R	217.44	95.47	90%	91%	96%
TM	**114.77**	**61.23**	**121.26**	**9.59**	**123.86**	**0.00**	**117.85**	**10.71**	**95%**	**93%**	**97%**

Table 2: Comparing results of BAC, ACO and PFSO for (n=200 to n=300).

n	**BAC**		**ACO**		**PFSO (K = 1)**		**PFSO (K = 3)**		**POP**		
	OP	**Time**	**BV**	**T**	**BV**	**T**	**BV**	**T**	**ACO**	**PFSO (K = 1)**	**PFSO (K = 3)**
200	NE	NE	235.74	90.84	254.08	R	229.90	173.82	98%	90%	Best
225	NE	*NE*	259.39	130.51	255.39	*R*	243.38	222.98	94%	95%	*Best*
250	*NE*	*NE*	272.10	170.95	266.60	*R*	253.60	332.66	93%	95%	*Best*
275	NE	NE	273.21	272.13	279.81	R	259.22	420.21	95%	93%	Best
300	NE	NE	295.91	478.17	289.49	R	274.48	718.70	93%	95%	Best
TM	**NE**	**NE**	**267.27**	**228.52**	**269.07**	**0.00**	**252.12**	**373.67**	**94%**	**94%**	**Best**

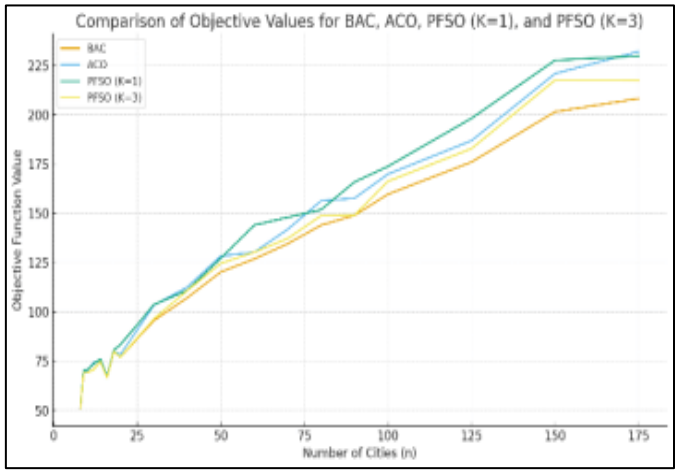


Figure 2: Comparison results of Table 1 f or $n = 8{:}175$.

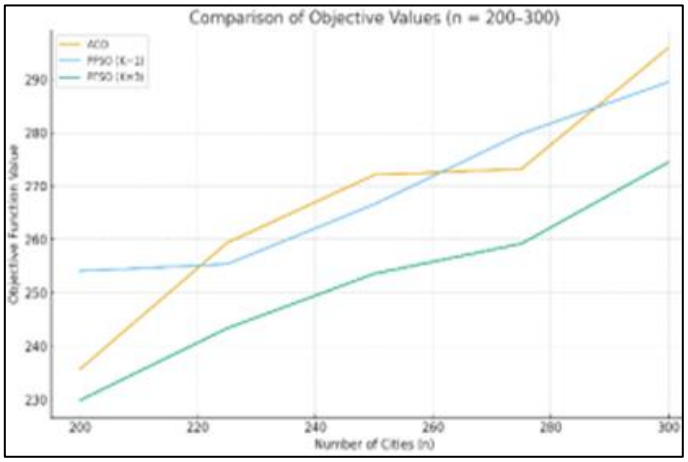


Figure 3: Comparison results of Table 2 for $n = 200{:}300$

Discussions and Analysis of Results

The experimental evaluation conducted on problem sizes ranging from 8 to 300 cities provides several important insights into the performance behavior of the proposed Peregrine Falcon Stoop Optimization (PFSO) method compared with the Ant Colony Optimization (ACO) algorithm. While the Branch & Cut method becomes infeasible beyond 175 cities, the extended results offer a clear picture of how PFSO (K=1) and PFSO (K=3) behave as the problem size increases.

From medium scall to large one, the dataset spectrum ($200 \leq n \leq 250$, in dollars), the PFSO (K=3) variant obtains a clear dominance, conforming the efficacy of the multi-pair swooping mechanism. At n=200 in dollars, for example, the algorithm pushes the objective cost to 229.90, reducing the ACO baseline (235.74) by almost 2.5% and passing the K=1 variant by 9.5%.

The difference of the advantage of K=3 structure becomes clear at the 300-city threshold, where structural differences generate stagnation. Thus, this method satisfies a value of 274.48 compared to ACO's 295.91.

This work comes directly from the harmony between expanded K-pair movement equations and a dynamic equilibrium between local correction and global restructuring.

Conclusion and Future Work

This article establishes a new meta-heuristic framework, Peregrine Falcon Swoop Optimization (PFSO), particularly, attached to address the scale challenges of the Traveling Salesman Problem (TSP). The algorithm was rigorously benchmarked through extensive numerical simulations, showing robust performance on complicated routing spaces extending to 300 cities. The test results show that PFSO can actually balance solution quality and cost by using its unique movement strategy based on K-pairs that imitate the activity of peregrine falcons. The K=1 case of PFSO shows stable competing performance for all sizes of the given problem, and the best performance is served for the K=3 case regarding the ant colony optimization algorithm passing by 5% to 10% improvement for all given sizes under test. The paper further examines PFSO high scale to high-dimensional search spaces in comparison with traditional granular optimization techniques. The dives involving multiple pairs to emulate peregrine falcons' simultaneous diving activity are demonstrated to be competent to make sufficient adjustments to the global route geometry without fully exploring the search space at the expense of runtime performance. This allows PFSO to obtain competitive runtime performance as well as considerable solution improvements over existing approaches by making sufficient adjustments to the global geometry of the solution trajectories without fully exploring the search space at the expense of runtime performance. Therefore, we can conclude that the PFSO algorithm is capable of solving the solution geometry of large-scale routing optimization problems independently. The PFSO algorithm itself can hence be concluded to be competent to effectively solve the solution geometry of large-scale routing optimization problems on its own. The K=3 version of the PFSO algorithm itself can hence be concluded to be competent to effectively solve the solution geometry of large-scale routing optimization problems on its own right. Some recommendations for further research were to generalize the PFSO algorithm to Mult objective routing optimization problems such as the Multi-Objective Traveling Salesman Problem (MOTSP), vehicle routing optimization problems involving vehicle location number optimization problems to effectively distribute vehicle traffic to reduce traffic congestion on roadways to improve traffic flow on roadways for increased traffic capacity to improve traffic efficiency on roadways to reduce health hazards that may arise from air emissions by vehicles due to traffic on the roadways to improve traffic safety on roadways to eliminate supplementary effort to navigate on roadways to improve vehicle efficiency on roadways to reduce vehicle traffic on roadways to improve traffic see for example [20,21,22,23].

References

[1] A. E. Abd Al Sada and B. A. Kalaf, “A Novel Hybrid WOFOA Algorithm for Efficient Multi-Objective APP Solutions,” AIP Conference Proceedings (2025).

[2] K. E. Abd Al Sada and B. A. Kalaf, “Solving the Integrated Production Planning and Scheduling Problem via a New Hybrid Meta-Heuristic Algorithm,” AIP Conference Proceedings (2025).

[3] X. Zhang, et al., “Multi-objective vehicle routing with time windows using a new hybrid algorithm,” Information Sciences, **660**, 120150 (2024).

[4] K. Deb and H. Jain, “Evolutionary Multi-Criterion Optimization for Large-Scale Problems,” IEEE Transactions on Evolutionary Computation, **18**(4) (Classic Ref).

[5] A. Bajaj and J. Dhodiya, “Sustainable multi-depot multiple travelling salesman problem by robust multi objective quasi oppositional jaya algorithm,” Journal of Industrial & Management Optimization, **21**(4), 2885-2924 (2025).

[6] L. Zhou, et al., “A survey of deep reinforcement learning for the traveling salesman problem,” Expert Systems with Applications, **241**, 122687 (2024).

[7] M. R. Ghaffari, et al., “Recent advances in nature-inspired algorithms for vehicle routing problems,” Artificial Intelligence Review, **57**, 1–45 (2024).

[8] S. Wang, et al., “Optimizing the multi-objective traveling salesman problem with a deep reinforcement learning algorithm using cross fusion attention networks,” Neural Networks, **192**, 107904 (2025).

[9] Z. Wang, H. Sun, and C. P. Chen, “Towards Generalizable Meta-Deep Reinforcement Learning Algorithm for Multi-Objective Traveling Salesman Problems,” IEEE Transactions on Artificial Intelligence, **5**(1), 1-14 (2024).

[10] L. Xin, et al., “Diversity Optimization for Travelling Salesman Problem via Deep Reinforcement Learning,” arXiv preprint arXiv:2501.00884 (2025).

[11] J. Yi, et al., “Combinatorial Optimization with Generative Sampling (COGS),” arXiv preprint arXiv:2508.08718 (2025).

[12] S. Mirjalili, et al., “Nature-inspired algorithms: A comprehensive review of the state-of-the-art,” Applied Soft Computing, **150**, 111099 (2024).

[13] A. G. Hussien, “Modern metaheuristic algorithms: A review of recent variants and applications,” Engineering Applications of Artificial Intelligence, **129**, 107624 (2024).

[14] E. Osaba, et al., “Bio-inspired optimization in 2024: Trends and future directions,” Swarm and Evolutionary Computation, **85**, 101480 (2024).

[15] Y. A. Mukhlif, et al., “Ant Colony and Whale Optimization Algorithms Aided by Neural Networks for Optimum Skin Lesion Diagnosis: A Thorough Review,” Mathematics, **12**(7), 1-29 (2024).

[16] M. Dorigo and T. Stützle, “Ant Colony Optimization: Overview and Recent Advances,” IEEE Computational Intelligence Magazine, **18**(4) (2023).

[17] Y. Ni, et al., “An Intelligently Enhanced Ant Colony Optimization Algorithm for Global Path Planning of Mobile Robots,” Sensors, **25**(5), 1326 (2025).

[18] R. Mills, G. K. Taylor, and L. C. Hemelrijk, “Physics-based simulations of aerial attacks by peregrine falcons reveal that stooping at high speed maximizes catch success against agile prey,” PLoS Computational Biology, **14**(4), e1006044 (2018).

[19] V. A. Tucker, “Gliding flight: speed and acceleration of ideal falcons during diving and pull out,” Journal of Experimental Biology, **201**(3), 403-414 (1998).

[20] Z. Zhang, et al., “Coordinated seru scheduling and distribution operation problems with DeJong's learning effects,” European Journal of Operational Research, **313**(2), 452-464 (2024).

[21] M. S. Aldhabayan, et al., “MoTSPPP: Multi-objective Traveling Salesman Problem with Profits and Passengers,” ResearchGate Preprint (2025).

[22] T. J. Khraibet, B. A. Kalaf, and W. Mansoor, “A new meta-heuristic algorithm for solving multi-objective single machine scheduling problems,” Journal of Intelligent Systems, **34**, 20240373 (2025).

[23] N. M. Neamah and B. A. Kalaf, “Solving multi-criteria scheduling: total completion time, late work, and maximum earliness,” Iraqi Journal of Science, **65**, 2724–2735 (2024).