Enhancing Grey Wolf Optimization Using Preprocessing Techniques to Solve the Travelling Salesman Problem with Time Windows

Archana A. Deshpande
G H Raisoni University, Amravati, Maharashtra, India
archana.deshpande@raisoni.net (corresponding author)

Seema Raut
Applied Mathematics Department, G H Raisoni University, Amravati, Maharashtra, India
seema.raut@raisoni.net

Nalini V. Vaidya
Applied Mathematics Department, G H Raisoni College of Engineering, Nagpur, Maharashtra, India
nalini.vaidya@raisoni.net

Received: 21 June 2025 | Revised: 31 July 2025, 22 August 2025, and 24 September 2025 | Accepted: 27 September 2025

 DOI: https://doi.org/10.48084/etasr.12852

ABSTRACT

The aim of the NP-hard Traveling Salesperson Problem with Time Windows (TSPTW) is to visit a predetermined customer group within the time frames allotted to them while minimizing a predetermined objective function. This problem considers a salesman who leaves his house, has to go to several places in a given amount of time, and then returns. The Gray Wolf Optimizer (GWO) is a bioinspired meta-heuristic population-based algorithm that mimics the survival strategies of gray wolves. This study applies the GWO strategy to minimize travel expenses within the allotted period, incorporating preprocessing to improve performance. The effectiveness of the proposed method is evaluated using reputable benchmark cases to reduce overall travel expenses. The MATLAB environment was used to implement the GWO. Based on the computational results, GWO performs much better than other similar algorithms.

Keywords-traveling salesman problem with time window; grey wolf optimization; preprocessing

I. INTRODUCTION

There are various variants of the Traveling Salesman Problem (TSP) [1], with the TSP with Time Windows (TSPTW) being the most significant variant. Similar to the conventional problem, TSPTW also demands the Hamiltonian circuit around a graph of vertices, but also stipulates that every node needs to be visited within a set time frame. TSPTW may arise in distribution logistics, including job-shop scheduling, school bus routing, industrial waste collection, bank and postal delivery, perishable commodity transportation, military operations, etc. TSPTW is a combinatorially complex optimization problem that has been proven to be NP-hard, which means that there is no known algorithm that can solve all instances of the problem in polynomial time. TSPTW involves the task of determining an appropriate route that visits several clients, each within specified time windows, with the beginning and ending at a specific depot, and is commonly used in routing applications [2]. To solve TSP, several precise techniques are available, such as the Lagrangian dual, branch and bound, and cutting plane [3]. Although these conventional techniques are effective in solving TSP, there are situations in which they are not appropriate, such as when the problem is too big, the data gathering is not precise, the goals of the problem clash, etc. Furthermore, due to the complexity of the problem, solving TSP with standard approaches requires a very large amount of computational time. To overcome these challenges, research efforts use heuristics and metaheuristics to solve TSP effectively.

Although metaheuristics were introduced long ago, their ability to find efficient solutions to optimization problems keeps them interesting today. The benefit of metaheuristic approaches over conventional approaches is that, in situations where traditional methods prove inadequate or ineffective, they can be used to identify a more practical and workable option.

In some real-world applications, seeking the best solution might be inappropriate due to a variety of reasons, such as the problem's magnitude, the difficulty of defining certain constraints, the existence of competing goals, the imprecision of gathering data, the changing nature of the workplace, etc. In certain situations, exact approaches either do not work to solve the problem or do not produce an appropriate solution. For such instances, metaheuristics typically yield an appropriate answer in a fair amount of computing time.

However, metaheuristics do not ensure a globally optimal solution as do iterative approaches and standard optimization techniques. The TSP is NP-hard since it becomes computationally challenging to solve in an acceptable period of time as the problem's size (the number of cities) grows. Nevertheless, TSP remains a key optimization problem, and different approaches, such as approximation algorithms or heuristics, are frequently employed to identify near-optimal solutions for large instances.

Metaheuristics are an essential component of routing problems and are currently a hot issue in science. It is challenging to examine every potential workable solution, especially to acquire the optimal TSP solution, particularly when the search area is large. In the last ten years, many algorithms based on biological systems, such as swarm intelligence, annealing processes, Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), etc., have been proposed to address these problems. In addition, numerous Particle Swarm Optimization (PSO) variants have been suggested to effectively solve TSP [4-6]. The Combinatorial ABC (CABC) algorithm was developed to address combinatorial optimization issues [7]. In a hybrid technique called GA-PSO-ACO [8], ACO is used for exploitation, while exploration is carried out by PSO and GA. In [9], ABC used k-Opt to solve TSP.

The Discrete Spider Monkey (DSM) algorithm is one of the most recent algorithms to effectively solve the TSP [10]. In recent years, other studies have proposed nature-inspired techniques, such as ACO [11], PSO [12], ABC [9, 13], the Firefly Algorithm (FA) [14], and others, to identify optimal solutions of the TSP. In order to obtain better results, in [15], a hybrid approach of ant colony system with PSO (GSA-ACS-PSOT) was proposed to solve the TSP. In [16], the Discrete Gray Wolf Optimizer (DGWO) was used for symmetric TSP. In [17], a hybrid approach of Firefly and ACO was employed to solve the multi-objective TSP.

In [18], a first attempt was made to solve the TSPTW optimally. Using dynamic programming, a branch-and-bound approach was proposed, which solved instances with up to 50 nodes optimally. In [19], the Beam-ACO and compressed annealing algorithms were adapted from the travel-time form of the TSPTW that took into account makespan optimization. In [20], a multifactorial evolutionary algorithm was employed to solve TSPTW, also investigating the Traveling Repairman Problem (TRPTW). In [21], an RL-based model was proposed to address the TSPTW. In order to accurately represent the joint distribution between the problem features and the solution, a variety of encoders and decoders were used in the generated model.

This study uses Gray Wolf Optimization (GWO) to solve the TSPTW. Numerous applications of GWO for engineering purposes, including design and controller tuning, image processing, robotics, power dispatch, Wireless Sensor Networks (WSNs), clustering, etc., are documented in the literature. In addition, researchers have presented numerous modifications to GWO in an effort to address real-world issues. In [22], a discrete variant of GWO was presented for the two-stage assembly flow shop scheduling problem with release time. In [23], GWO was used to plan the safe route of an unmanned combat aerial vehicle while using the least amount of fuel possible. In [24], GWO was used to solve a non-convex, discontinuous economic dispatch problem. For multiple image thresholding, a Modified DGWO (MDGWO) was presented in [25], which improved the gray wolves' optimal solution updating mechanism by using weights.

Recently, in [26], GWO was applied to resolve the hybrid flow shop scheduling problem, which has received extensive study. To address this issue, this study suggested the Multi-Objective Cellular Grey Wolf Optimizer (MOCGWO) and employed the variable neighborhood search technique. Using wrapper approaches to merge GWO with two-phase mutations, in [27], GWO was used to address feature selection for classification problems. In [28], a hybrid multi-objective GWO was presented for dynamic scheduling in the actual welding sector, presenting a modified social hierarchy to enhance the GWO's capacity for exploration and exploitation. In [29], GWO was used to analyze online product reviews with deep learning based sentiment analysis.

II. MATHEMATICAL MODELLING

This study investigates a scenario in which a salesman leaves from the depot, the starting point, travels to every place in a predetermined set only once, and then returns to the depot. TSPTW is the name given to the issue of selecting a journey, or group of such moves, in given time windows. This study makes the widely-held assumption that travel times follow the First-In First-Out (FIFO) rule, which states that a later departure cannot result in an early landing when traveling between two places, and that the time of travel is a linear function of the time of departure. The proposed approach starts by trying to reduce a makespan objective that searches for a tour that can be finished to return to the starting point as soon as possible after the planning horizon starts. The FIFO property allows for an optimal solution where the vehicle waits, if at all, only when it reaches a site before the time window for that location opens.

This study discusses the approach to best achieve a duration goal, which is to finish the trip as soon as possible after leaving the depot. To accomplish this objective, the depot departure time is adjustable so that the total tour time may be reduced by leaving the depot at a later time than the scheduled horizon's beginning.

Let $G = (V, E)$ be a directed graph with $V = \{0, 1, 2, \dots, l\}$ a collection of the nodes or cities, depot is 0, and $E = \{(i, j), i \neq j;\ i, j \in V\}$ be the set of edges. There is a time frame $[p_i, q_i]$ that needs to be visited for each location $i = 1, 2, \dots, l$. It should be noted that the vehicle might get to the point $i \in V_0$ before the time window begins, where in this case the vehicle

will have to wait till the window opens. Additionally, the depot has a time frame $[p_0, q_0]$ that specifies that the vehicle cannot leave the depot before p_0 and cannot return there after q_0.

Arcs that indicate motion between locations make up the set $E \in V \times V$. A piecewise linear travel time function $T_{ij}(t)$ is connected to every edge (i,j) in E and provides the time of travel along the edge (i,j) if the tour of the arc starts at time t. Every tour duration is a finite piecewise linear function that adheres to the FIFO principle, which means that for every $(i,j) \in E$ and times t and t', $t \leq t'$, such that $t + T_{ij}(t) \leq t' + T_{ij}(t')$. Formally speaking, the TSPTW can be solved by a succession of node-time pairs, such as $(i_0, t_0), (i_1, t_1), \dots, (i_l, t_l), (i_{l+1}, t_{l+1})$ where $i_0 = i_{l+1} = 0$, $\{i_1, i_2, \dots, i_l\} = \{1,2, \dots, l\}$ $t_0 \geq p_0$, $t_{k+1} \geq max\{p_{k+1}; t_k + T_{i_k,i_{k+1}}(t_k)\}$ $\forall k = 0, \dots, n$, $t_{l+1} \leq q_0$. While the duration goal seeks to minimize $t_{l+1} - t_0$, the makespan objective seeks to minimize t_{l+1}.

For every edge $\big((i;\, t);\, (j;\, t')\big)$, the binary variable $y((i;t);(j;\, t'))$ is defined to represent whether a salesman follows that edge or not. This study uses the following modelling of the TSPTW.

$$Z = minimize \sum_{((i,t),(j,t'))} C_{ij} y_{((i,t),(i,t'))} \quad (1)$$

$$subject\ to \sum_{((i,t),(j,t'))} y_{((i,t),(i,t'))} = 1 \quad (2)$$

$$\sum_{((i,t_1),(j,t_2))} y_{((i,t_1),(j,t_2))} - \sum_{((j,t_3),(i,t_2))} y_{((j,t_3),(i,t_2))} = 0 \quad (3)$$

$$\sum_{((i,t_1),(j,t_2))} y_{((i,t_1),(j,t_2))} \in \{0,1\} \quad (4)$$

Constraint (2) guarantees that the vehicle reaches each place precisely once inside its time frame; Constraint (3) guarantees that the salesman leaves each city at which it arrives (apart from the depot), and constraint (4) specifies the domains of the decision factors.

III. GREY WOLF OPTIMIZER (GWO) ALGORITHM

GWO is a relatively recent population-based algorithm that draws inspiration from the leadership structure and hunting tactics of gray wolves. Gray wolves typically coexist in a rigid, structured group. Alpha, beta, delta, and omega are the four varieties of gray wolves that practice hanging, encircling, and attacking. The best answer is represented by alpha, followed by beta, delta, and finally omega. The alpha, beta, and delta positions are the main guiding locations used by gray wolves. They split off to hunt and then come back together to attack their prey. When the search agent may be anywhere between the prey's site and the current site, the algorithm handles the optimization difficulties, as shown by its pseudo-code in Algorithm 1. The gray wolf is one of the top predators. The following three categories describe their hunting behavior.

- Tracing and moving toward the target
- Surrounding and frightening the target
- Attacking the target

The following mathematical model can be used to represent the wolves' surrounding strategy around their prey:

$$C = 2\, Random2 \quad (5)$$

$$B = 2 \cdot b \cdot Random1 - b \quad (6)$$

where b decreases linearly from 2 to 0, and $Random1$ and $Random2$ have evenly distributed random numbers between 0 and 1. Over iterations, b is expressed as

$$b = 2 - s * \frac{2}{N} \quad (7)$$

where N is max iterations and:

$$D = |C \cdot XP(s) - X(s)|$$

$$X(s+1) = |XP(s) - B \cdot D|$$

Using the location of prey $X(s)$ at s iteration and D as the difference vector, $X(s+1)$ is the location of the wolf at the $(s+1)$ iteration.

Every wolf utilizes the potential of alpha, beta, and delta wolves in searching tactics since they are the most efficient in a group. By using alpha, beta, and delta, the wolves improve their location as follows:

$$X_1 = |X_\alpha - B_\alpha \cdot D_\alpha|, \text{ where } D_\alpha = |C \cdot X_\alpha - X|$$

$$X_2 = |X_\beta - B_\beta \cdot D_\beta|, \text{ where } D_\beta = |C \cdot X_\beta - X| \quad (8)$$

$$X_3 = |X_\delta - B_\delta \cdot D_\alpha|, \text{ where } D_\delta = |C \cdot X_\delta - X|$$

$$X(s+1) = \frac{X_1 + X_2 + X_3}{3} \quad (9)$$

where X_α, X_β, and X_δ represent the alpha wolves', beta wolves', and delta wolves' approximate locations. The updated position of the wolf is shown in (9). The GWO parameters C and B balance exploration and exploitation.

```
Algorithm 1
Set the grey wolf population X_i where i
varies from 1 to l
1. Assign initial values to the variables
   b, B, C
2. Determine the fitness level of every
   wolf. The optimal search agent is
   represented by X_α, the second optimal
   search agent is represented by X_β, the
   third optimal search agent is
   represented by X_δ .
3. For all wolves, while (s < the max
   number of iterations), change the
   positions of the search agents using
   (8).
4. Update b, B, and C and evaluate the
   fitness values for all search agents
5. Renew X_α, X_β, and X_δ
6. Increase i by 1
7. End while
Output
```

IV. PROPOSED ALGORITHM

This section describes the proposed GWO for resolving the TSPTW. Before that, preprocessing is used to obtain better solutions.

Preprocessing is used to both tighten time windows and prune arcs. In [30], a new approach was proposed to solve the Time Dependent TSPTW (TD-TSPTW). The methods employed here involve calculating the numbers $\Delta_{ij}(t)$, indicate the earliest possible time for the vehicle to leave site j, presuming that it has already left place i at time t. In cases where journey times meet the triangle inequality, $\Delta_{ij}(t)$ can be set as:

$$\Delta_{ij}(t) = max\{p_j;\ t + T_{ij}(t)\} \quad (10)$$

If not, these Δ values can be determined by figuring out the shortest paths; however, it should be noted that the validity of those Δ values depends on the FIFO characteristic. First, rules are introduced to update time frames at locations using these Δ values. All TSPTW and TD-TSPTW variations are covered by the principles that are presented. The regulation establishes that the salesman cannot depart k before it arrives from another location.

$$p_k = max\{p_k;\ min\{q_k; \min i \in N, k\ \Delta_{ij}(p_i)\}\} \quad (11)$$

The second rule sets

$$q_k = min\{q_k;\ max\{p_k;\ max\ i \in N, k\ max\ t\ \Delta_{ki}(t) \leq q_i\}\} \quad (12)$$

acknowledging that the salesman can leave a point at times that allow him to get to a different location within its time window. To reduce the number of arcs from E, the time windows at two different locations are examined; For instance, the sequence $i \rightarrow j$ is marked unfeasible if there is no moment at which the salesman can leave from i to j and reach inside the time limit of j. Testing to see if $\Delta_{ij}(e_i) > j$ makes checking this simple because of the FIFO property.

As a result, it can be stated that j must be visited before i, which is represented by $j \prec i$. Using this precedence relationship, (i,j) is removed for A. In a similar way, the connections between the time periods are examined at three different places. Arc (i,j) can be removed if it is not possible to go in the order $i \rightarrow j \rightarrow k$ and $k \rightarrow i \rightarrow j$. If the sequence of $i \rightarrow k \rightarrow j$ is also impractical, it can be concluded that $j \prec i$. Lastly, it can be seen that precedence is transitive, meaning that it can determine that $i \prec k$ if $i \prec j$ and $j \prec k$. Instead of removing the above arc, a new variable is introduced

$$slack_{i,j} = q_j - \Delta_{ij}(e_i)$$

where:

- If $slack_{i,j} < 0$, the arc is infeasible (remove the arc)
- If $0 \leq slack_{i,j} < \in$, the arc is considered with low priority

Algorithm 2 describes the preprocessing technique to delete infeasible arcs.

```
Algorithm 2
Input: Graph G = (V; E) and time windows
       [p_i, q_i] for all i ∈ N
Output: Set U with a shorter route
1. Set U = {}, denoting the set of known
   precedence relationships.
2. If changes are found, do
3. For each edge (i,j) ∈ E:
     Compute Δ_ij(e_i) = max(p_j, p_i + T_ij)
     Compute Slack_ij = q_j − Δ_ij(e_i)
     If Slack_ij < 0
       i → j is infeasible
       Add precedence j ≺ i to U
     Else if 0 ≤ Slack_ij <∈
       (i,j) is almost infeasible,
       optionally mark (i,j) as weak (low
       priority arc)
     For all triplets (i,j,k) ∈ V:
     If all of the following are
     infeasible:
       i → j → k
       k → i → j
       i → k → j
      Then
        Add j ≺ i to U
     If both i → j → k and k → i → j are
     infeasible:
       Delete arc (i,j) from E
   End
4. To ensure that U has all precedence,
   add a transitive closure of precedence
   to it.
   if i ≺ j and j ≺ k then i ≺ k
   Delete each edge (j,i) from E if edge
   (i,j) ∈ U
   Delete each edge (i,k) from E if there
   is j such that (i,j), (j,k) ∈ U.
   Delete each edge (i,0) and (0,j) from E if
   (i,j) ∈ U
   End
```

After preprocessing, the algorithm creates the initial feasible solution. The wolf i completes D_α preprocessing executions during each cycle, and likewise D_β and D_δ. Thus, each wolf updates its position as follows by using this approximation:

$$X_1 = preprocessing\ (X_i, D_\alpha)$$

$$X_2 = preprocessing\ (X_i, D_\beta)$$

$$X_3 = preprocessing\ (X_i, D_\delta) \quad (13)$$

$$X = \frac{X_1 + X_2 + X_3}{3} \quad (14)$$

GWO was first proposed for continuous optimization issues, as was previously mentioned. In [16], changes were

made to the original GWO to deal with the Discrete TSP. In the proposed algorithm, time windows with preprocessing are used to update GWO for TWTSP. Preprocessing speeds up the optimization process and enhances the quality of the final solution by ensuring that the GWO algorithm concentrates on feasible options and avoids exploring invalid solutions.

Every grey wolf in the swarm stands for a workable TSP solution within the allotted time range. This algorithm considers finding a path that minimizes the salesman's cost time as one of its objectives. The steps involved in the algorithm are shown in Figure 1.

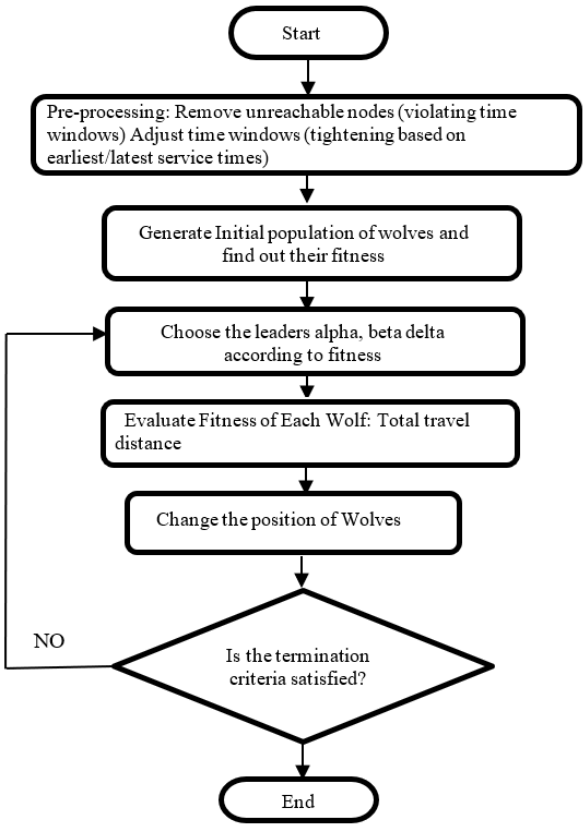

Fig. 1. Flow of the proposed algorithm.

Algorithm 3
1. Parameter Initialization:
Assign the number of salesmen,
Number of cities,
Time window of each city
2. Use the preprocessing technique:
Remove unreachable nodes (violating time windows); Adjust time windows (tightening based on the earliest/latest service times)
$slack_{i,j} = q_j - \Delta_{ij}(e_i)$
If $slack_{i,j} < 0$, the arc is infeasible.
If $0 \leq slack_{i,j} < \varepsilon$, the arc is almost infeasible.
3. Calculate the values as per (5),(6),(7),(8)
4. Assign the number of wolves for every salesman.
Discover where each salesman should start, determine each salesman's initial move within the time range.
For all salesmen
5. Determine which cities to visit next using (13)
6. Determine each agent's fitness level
Use (14) for updating the current search agent's position.
7. Determine his tour
$Number\ of\ city = Number\ of\ city - 1$
If $Number\ of\ city$ is greater than 0
Determine the salesman who starts first, then use (12) to determine the next city.
8. Change position using (13)
9. End if – Return to depot
For $i = 1$ to $Number\ of\ salesmen$
For $j = 1$ to $Number\ of\ the\ cities$
10. Change the position using (14)
End for
Output: The best possible path for every salesman

V. EXPERIMENTAL RESULT

The proposed GWO algorithm was evaluated on some TSP examples, which vary from 20 to 240 cities. GWO experiments were carried out on a MATLAB-R2015a on an individual laptop with an Intel Core i5 8250U 1.80G Hz processor and 4.00 GB of RAM. Random permutation was used to create the initial population for each of the heuristics. Additionally, comparable settings were applied to all algorithms, including TSP benchmarks, to ensure a fair comparison. The proposed algorithm used a preprocessing technique and 100 gray wolves in the initial population. However, to simplify the process, some of the original GWO parameters were ignored. After using the preprocessing algorithm repeatedly to remove infeasible solutions, GWO was used to obtain an improved solution. The proposed algorithm was compared with the best solutions taken from published works [19, 20, 31, 32, 33], as shown in Tables I, II, and III. Figure 2 shows the convergence graphs for some TSP instances. Table I shows the comparative analysis for optimal results from instances in [19, 31]. Table II shows results for instances from [19, 32], and Table III shows results for examples in [20, 31] and for symmetric instances in [33]. To compare the obtained optimal solution with known solutions, the gap between solutions was calculated using:

$$Gap = Optimal\ Solution\ from\ the\ proposed\ algorithm - Known\ Optimal\ solution$$

and Gap percentage. The average run time for the TSP instance was 1.76764 s.

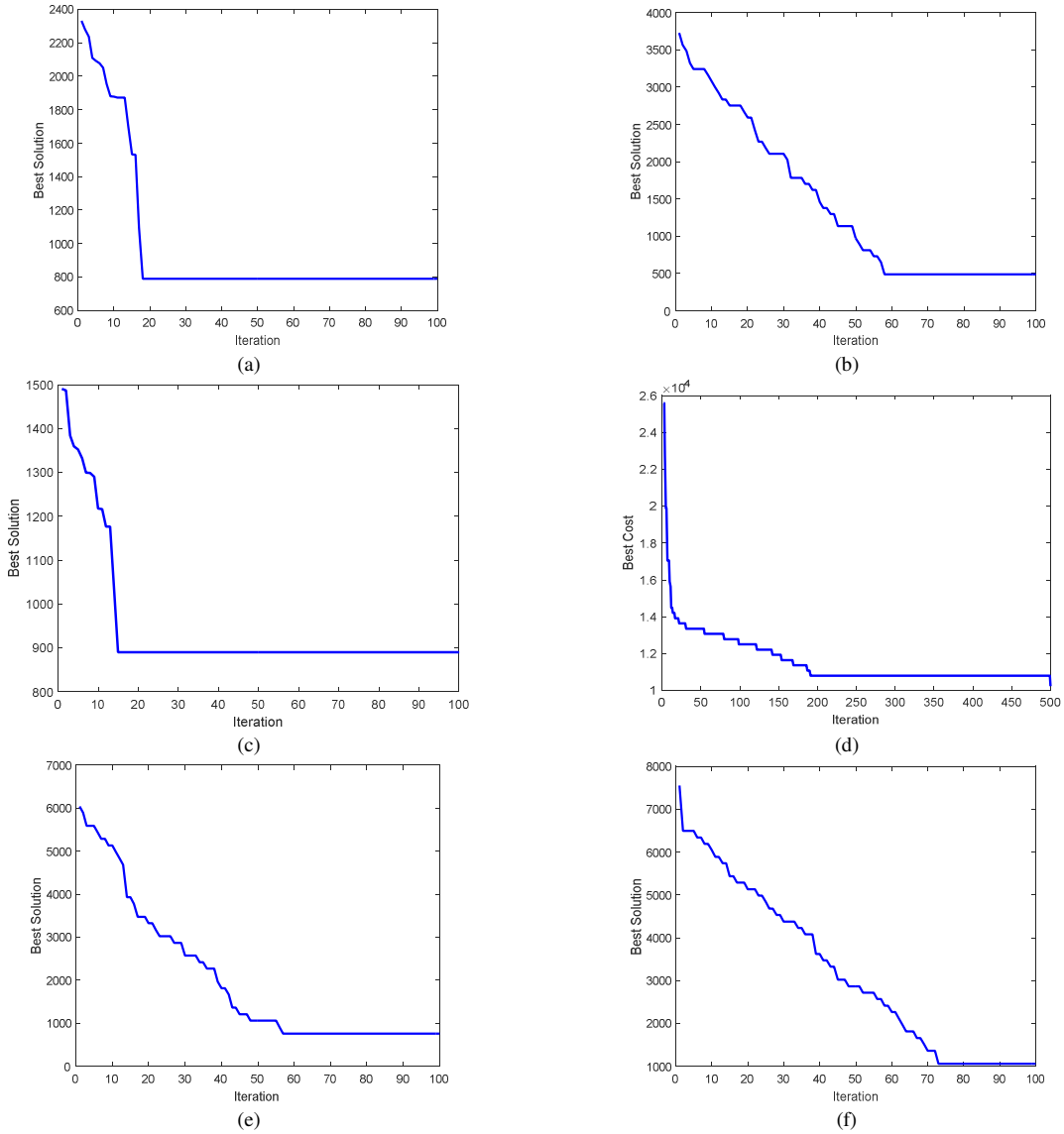

Fig. 2. Convergence curves for TSP instances: (a) rc_207.1, (b) n80w160.3, (c) rc_205.2, (d) n200w120.001, (e) n150w120.003, (f) n150w160.001.

TABLE I. COMPARISON WITH RESULTS FROM [19, 31]

TSP instances	Optimal from [19, 31]	Optimal from the proposed algorithm	Gap	% Gap
n200w120.001	1089	1080	-9	-0.83333333
n200w120.002	1072	1080	8	0.740740741
n200w120.003	1128	1137	9	0.791556728
n200w120.004	1072	1080	8	0.740740741
n200w120.005	1073	1080	7	0.648148148
n150w120.001	972	982	10	1.018329939
n150w120.002	917	927	10	1.078748652
n150w120.003	-	907		-
n150w120.005	925	937	12	1.280683031
n150w160.001	907	907	0	0
n150w160.004	-	1058		-

TABLE II. COMPARISON WITH RESULTS FROM [19, 32]

TSP instances	Optimal from [19, 32]	Optimal from the proposed algorithm	Gap	% Gap
rc208.2	579.51	574.0559	-5.4541	-0.950099111
rc208.1	810.7	792.7634339	-17.9365660	-2.262537008
rc207.1	804.67	788.36	-16.31	-2.068851794
rc206.1	756.45	756.45	0	0
rc206.2	870.49	860.1854998	-10.3045001	-1.197939303
rc205.2	820.19	812.016	-8.174	-1.006630411
rc204.2	870.52	841.3355887	-29.1844113	-3.468819307
rc204.3	455.03	447.45	-7.58	-1.694044027
rc203.3	921.44	945.45	24.01	2.53953144
rc203.2	853.71	850.26	-3.45	-0.405758239
rc202.2	938.52	917.7881153	-20.7318847	-2.258896622
rc202.3	894.1	892.52	-1.58	-0.177026845
rc201.4	889.18	892.52	3.34	0.374221306
rc201.3	722.43	725.93	3.5	0.482140151
rc201.2	860.17	859.4	-0.77	-0.089597394
rc201.1	592.06	585.8690809	-6.19091908	-1.056706915

TABLE III. COMPARISON WITH RESULTS FROM [20, 31] AND FINDINGS FOR SYMMETRIC EXAMPLES IN [20, 33]

TSP instances	Known Optimal from [20, 31] and [20, 33]	Optimal from the proposed algorithm	Gap	% Gap
n80w120.002	577	568	-9	-1.584507042
n80w120.003	540	547	7	1.279707495
n80w140.2	472	487	15	3.080082136
n100w120.2	540	506	-34	-6.719367589
n80w160.3	521	487	-34	-6.981519507
n150w120.3	747	756	9	1.19047619
n150w140.1	762	756	-6	-0.793650794
n150w160.2	711	790	79	10.0000000

TABLE IV. PAIRED T- TEST ON TSP INSTANCES

Proposed algorithm	TSP instances shown in Table I	TSP instances shown in Table II	TSP instances shown in Table III
Standard deviation	6.584915	28.54539	12.10191381
Degrees of Freedom	8	7	15
t- value	2.784141	2.018862	2.000022124
p-value	0.01189	0.04164	0.03197

VI. CONCLUSION

This study presented an innovative technique for solving TSPTW using GWO and a preprocessing technique. The motivation behind using the preprocessing technique is to remove arcs that are either too costly or too inconveniently included in a complete graph. The combination of metaheuristic optimization and an efficient preprocessing step, which reduces the number of impractical pathways and improves the quality of the solutions, is what makes this approach different. In time-constrained routing scenarios, this combination technique outperforms classic GWO applications in terms of convergence speed and solution feasibility. The amount of time required to compute a satisfactory solution is reduced by using preprocessing. The optimization process is streamlined by preprocessing stages, which eliminate duplication and direct the search toward more practical options. Thus, the preprocessing stage before applying GWO helps to obtain faster and better solutions. The proposed method was evaluated on 27 TSP instances. Regarding both the computation time and the accuracy of the solutions, the proposed method yielded comparable results. In certain cases, the proposed method yielded superior outcomes, but in some other cases, it was the other way around. Future work will investigate modifications to the algorithm and apply them in real-life applications.

REFERENCES

[1] S. Bock, S. Bomsdorf, N. Boysen, and M. Schneider, "A survey on the Traveling Salesman Problem and its variants in a warehousing context," *European Journal of Operational Research*, vol. 322, no. 1, pp. 1–14, Apr. 2025, https://doi.org/10.1016/j.ejor.2024.04.014.

[2] R. Fontaine, J. Dibangoye, and C. Solnon, "Exact and anytime approach for solving the time dependent traveling salesman problem with time windows," *European Journal of Operational Research*, vol. 311, no. 3, pp. 833–844, Dec. 2023, https://doi.org/10.1016/j.ejor.2023.06.001.

[3] P. C. Pop, O. Cosma, C. Sabo, and C. P. Sitar, "A comprehensive survey on the generalized traveling salesman problem," *European Journal of Operational Research*, vol. 314, no. 3, pp. 819–835, May 2024, https://doi.org/10.1016/j.ejor.2023.07.022.

[4] K. P Wang, L. Huang, C. G. Zhou, and W. Pang, "Particle swarm optimization for traveling salesman problem," in *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, 2003, pp. 1583–1585, https://doi.org/10.1109/ICMLC.2003.1259748.

[5] M. A. H. Akhand, S. Akter, and M. A. Rashid, "Velocity Tentative Particle Swarm Optimization to solve TSP," in *2013 International Conference on Electrical Information and Communication Technology (EICT)*, Feb. 2014, pp. 1–6, https://doi.org/10.1109/EICT.2014.6777868.

[6] X. Wei, Z. Jiang-Wei, and Z. Hon-lin, "Enhanced Self-Tentative Particle Swarm Optimization Algorithm for TSP," *Journal of North China Electric Power University,* vol. 36, no. 6, pp. 69–74, 2009.

[7] D. Karaboga and B. Gorkemli, "A combinatorial Artificial Bee Colony algorithm for traveling salesman problem," in *2011 International Symposium on Innovations in Intelligent Systems and Applications*, June 2011, pp. 50–53, https://doi.org/10.1109/INISTA.2011.5946125.

[8] W. Deng, R. Chen, B. He, Y. Liu, L. Yin, and J. Guo, "A novel two-stage hybrid swarm intelligence optimization algorithm and application," *Soft Computing*, vol. 16, no. 10, pp. 1707–1722, Oct. 2012, https://doi.org/10.1007/s00500-012-0855-z.

[9] I. Khan and M. K. Maiti, "A swap sequence based Artificial Bee Colony algorithm for Traveling Salesman Problem," *Swarm and Evolutionary Computation*, vol. 44, pp. 428–438, Feb. 2019, https://doi.org/10.1016/j.swevo.2018.05.006.

[10] M. A. H. Akhand, S. I. Ayon, S. A. Shahriyar, N. Siddique, and H. Adeli, "Discrete Spider Monkey Optimization for Travelling Salesman Problem," *Applied Soft Computing*, vol. 86, Jan. 2020, Art. no. 105887, https://doi.org/10.1016/j.asoc.2019.105887.

[11] Y. Wang and Z. Han, "Ant colony optimization for traveling salesman problem based on parameters optimization," *Applied Soft Computing*, vol. 107, Aug. 2021, Art. no. 107439, https://doi.org/10.1016/j.asoc.2021.107439.

[12] M. Clerc, "Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem," in *New Optimization Techniques in Engineering*, G. C. Onwubolu and B. V. Babu, Eds. Springer, 2004, pp. 219–239.

[13] A. H. Alaidi, S. D. Chen, and Y. Weng Leong, "Artificial Bee Colony with Crossover Operations for Discrete Problems," *Engineering, Technology & Applied Science Research*, vol. 12, no. 6, pp. 9510–9514, Dec. 2022, https://doi.org/10.48084/etasr.5250.

[14] M. Li, J. Ma, Y. Zhang, H. Zhou, and J. Liu, "Firefly algorithm solving multiple traveling salesman problem," *Journal of Computational and Theoretical Nanoscience*, vol. 12, no. 7, pp. 1277–1281, 2015.

[15] S. M. Chen and C. Y. Chien, "Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques," *Expert Systems with Applications*, vol. 38, no. 12, pp. 14439–14450, Nov. 2011, https://doi.org/10.1016/j.eswa.2011.04.163.

[16] K. Panwar and K. Deep, "Discrete Grey Wolf Optimizer for symmetric travelling salesman problem," *Applied Soft Computing*, vol. 105, July 2021, Art. no. 107298, https://doi.org/10.1016/j.asoc.2021.107298.

[17] A. A. Deshpande, S. Raut, and N. V. Vaidya, "Solving the Multi-objective Travelling Salesman Problem by an Amalgam of Fruit Fly Optimization and Ant Colony Optimization," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 15564–15569, Aug. 2024, https://doi.org/10.48084/etasr.7353.

[18] N. Christofides, A. Mingozzi, and P. Toth, "State-space relaxation procedures for the computation of bounds to routing problems," *Networks*, vol. 11, no. 2, pp. 145–164, 1981, https://doi.org/10.1002/net.3230110207.

[19] M. López-Ibáñez, C. Blum, J. W. Ohlmann, and B. W. Thomas, "The travelling salesman problem with time windows: Adapting algorithms from travel-time to makespan optimization," *Applied Soft Computing*,

vol. 13, no. 9, pp. 3806–3815, Sept. 2013, https://doi.org/10.1016/j.asoc.2013.05.009.

[20] H. B. Ban and D. H. Pham, "Solving optimization problems simultaneously: the variants of the traveling salesman problem with time windows using multifactorial evolutionary algorithm," *PeerJ Computer Science*, vol. 9, Jan. 2023, Art. no. e1192, https://doi.org/1p0.7717/peerj-cs.1192.

[21] M. Alharbi, A. Stohy, M. Elhenawy, M. Masoud, and H. El-Wahed Khalifa, "Solving Traveling Salesman Problem with Time Windows Using Hybrid Pointer Networks with Time Features," *Sustainability*, vol. 13, no. 22, Nov. 2021, Art. no. 12906, https://doi.org/10.3390/su132212906.

[22] G. M. Komaki and V. Kayvanfar, "Grey Wolf Optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time," *Journal of Computational Science*, vol. 8, pp. 109–120, May 2015, https://doi.org/10.1016/j.jocs.2015.03.011.

[23] S. Zhang, Y. Zhou, Z. Li, and W. Pan, "Grey wolf optimizer for unmanned combat aerial vehicle path planning," *Advances in Engineering Software*, vol. 99, pp. 121–136, Sept. 2016, https://doi.org/10.1016/j.advengsoft.2016.05.015.

[24] T. Jayabarathi, T. Raghunathan, B. R. Adarsh, and P. N. Suganthan, "Economic dispatch using hybrid grey wolf optimizer," *Energy*, vol. 111, pp. 630–641, Sept. 2016, https://doi.org/10.1016/j.energy.2016.05.105.

[25] L. Li, L. Sun, J. Guo, J. Qi, B. Xu, and S. Li, "Modified Discrete Grey Wolf Optimizer Algorithm for Multilevel Image Thresholding," *Computational Intelligence and Neuroscience*, vol. 2017, no. 1, 2017, Art. no. 3295769, https://doi.org/10.1155/2017/3295769.

[26] C. Lu, L. Gao, Q. Pan, X. Li, and J. Zheng, "A multi-objective cellular grey wolf optimizer for hybrid flowshop scheduling problem considering noise pollution," *Applied Soft Computing*, vol. 75, pp. 728–749, Feb. 2019, https://doi.org/10.1016/j.asoc.2018.11.043.

[27] M. Abdel-Basset, D. El-Shahat, I. El-henawy, V. H. C. de Albuquerque, and S. Mirjalili, "A new fusion of grey wolf optimizer algorithm with a two-phase mutation for feature selection," *Expert Systems with Applications*, vol. 139, Jan. 2020, Art. no. 112824, https://doi.org/10.1016/j.eswa.2019.112824.

[28] C. Lu, L. Gao, X. Li, and S. Xiao, "A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry," *Engineering Applications of Artificial Intelligence*, vol. 57, pp. 61–79, Jan. 2017, https://doi.org/10.1016/j.engappai.2016.10.013.

[29] D. Elangovan and V. Subedha, "Adaptive Particle Grey Wolf Optimizer with Deep Learning-based Sentiment Analysis on Online Product Reviews," *Engineering, Technology & Applied Science Research*, vol. 13, no. 3, pp. 10989–10993, June 2023, https://doi.org/10.48084/etasr.5787.

[30] N. Boland, M. Hewitt, M. Savelsbergh, and D. M. Vu, "Solving Time Dependent Traveling Salesman Problems with Time Windows." Optimization Online, May 30, 2018.

[31] J. W. Ohlmann and B. W. Thomas, "A Compressed-Annealing Heuristic for the Traveling Salesman Problem with Time Windows," *INFORMS Journal on Computing*, vol. 19, no. 1, pp. 80–90, Feb. 2007, https://doi.org/10.1287/ijoc.1050.0145.

[32] G. Pesant, M. Gendreau, J. Y. Potvin, and J. M. Rousseau, "An Exact Constraint Logic Programming Algorithm for the Traveling Salesman Problem with Time Windows," *Transportation Science*, vol. 32, no. 1, pp. 12–29, Feb. 1998, https://doi.org/10.1287/trsc.32.1.12.

[33] R. F. da Silva and S. Urrutia, "A General VNS heuristic for the traveling salesman problem with time windows," *Discrete Optimization*, vol. 7, no. 4, pp. 203–211, Nov. 2010, https://doi.org/10.1016/j.disopt.2010.04.002.