UNIVERSIDAD NACIONAL DE SAN MARTÍN

RCSI
https://revistas.unsm.edu.pe/index.php/rcsi
e-ISSN: 2709-992X

Original article
Artículo orginal
Jul-Dec, 2024

Enhancing the multiple traveling salesman problem solutions through Harris Hawks optimization and machine learning techniques

Mejora de las soluciones del problema del viajante múltiple mediante técnicas de aprendizaje automático y optimización de Harris Hawks

Hussein, Ahmed Abdulmunem[1]*
Hameed, Musa A.[2]
Ahmed, Saddam Hamdan[3]

[1]University of Samarra, Salah Al-Din, Iraq.
[2]College of Petroleum Process Engineering, Tikrit University, Salah Al-Din, Iraq.
[3]Ministry of Education, Directorate of Education Diyala, Diyala, Iraq.

Received: 19 Abr. 2024 | **Accepted:** 03 Jul. 2024 | **Published:** 10 Jul. 2024

Corresponding author*: ahmed.abd@uosamarra.edu.iq

How to cite this article: Hussein, A. A., Hameed, M. A. & Ahmed, S. H. (2024). Enhancing the multiple traveling salesman problem solutions through Harris Hawks optimization and machine learning techniques. *Revista Científica de Sistemas e Informática, 4*(2), e745. https://doi.org/10.51252/rcsi.v4i2.745

ABSTRACT

This work introduces an approach to solving the Multiple Traveling Salesman Problem (mTSP) by integrating metaheuristic algorithms (MHs) with machine learning (ML) techniques. Specifically, the Discrete Harris Hawks Optimization (DHHO) algorithm was developed to handle the discrete nature of the mTSP, as the original Harris Hawks Optimization (HHO) was designed for continuous problems. The DHHO algorithm, enhanced with SARSA-based learning mechanisms for solution initialization and parameter tuning, significantly improves the efficiency of mTSP solutions. By leveraging ML's adaptability within the robust MH framework, this study offers a novel perspective on combinatorial optimization problems, surpassing the best-known solutions (BKS) in various mTSP instances. The results were tested using TSPLIB benchmark instances, including Att48, Berlin52, Bier127, Pr76, and Rat99, for two, three, and four salesmen, achieving optimal results in 12 out of 15 instances. The DHHO's performance was validated by the quality of solutions and consistency across multiple runs, with optimal results in 5 out of 5 instances for two salesmen, 3 out of 5 for three salesmen, and 4 out of 5 for four salesmen. Statistical validation using the Wilcoxon signed-rank test confirmed the significance of these improvements ($p < 0.05$). This work highlights the impact of integrating MHs and ML, making a substantial contribution to the current literature.

Keywords: combinatorial optimization problems; initialization; metaheuristic; parameter tuning; reinforcement learning

RESUMEN

Este trabajo presenta un enfoque para resolver el Problema del Viajante Múltiple (mTSP) mediante la integración de algoritmos metaheurísticos (MHs) con técnicas de aprendizaje automático (ML). En particular, se desarrolló el algoritmo de Optimización Discreta de Halcones de Harris (DHHO) para manejar la naturaleza discreta del mTSP, ya que el algoritmo original de Optimización de Halcones de Harris (HHO) fue diseñado para problemas continuos. El algoritmo DHHO, mejorado con mecanismos de aprendizaje basados en SARSA para la inicialización de soluciones y ajuste de parámetros, mejora significativamente la eficiencia de las soluciones del mTSP. Al aprovechar la adaptabilidad del ML dentro del robusto marco de MH, este estudio ofrece una nueva perspectiva sobre los problemas de optimización combinatoria, superando las mejores soluciones conocidas (BKS) en varias instancias del mTSP. Los resultados se probaron utilizando instancias de referencia de TSPLIB, incluyendo Att48, Berlin52, Bier127, Pr76 y Rat99, para dos, tres y cuatro vendedores, logrando resultados óptimos en 12 de las 15 instancias. El rendimiento del DHHO se validó por la calidad de las soluciones y la consistencia a lo largo de múltiples ejecuciones, obteniendo resultados óptimos en 5 de 5 instancias para dos vendedores, 3 de 5 para tres vendedores y 4 de 5 para cuatro vendedores. La validación estadística mediante la prueba de rango con signo de Wilcoxon confirmó la significancia de estas mejoras ($p < 0.05$). Este trabajo destaca el impacto de integrar MHs y ML, contribuyendo de manera sustancial a la literatura actual.

Palabras clave: problemas de optimización combinatoria; inicialización; metaheurísticas; ajuste de parámetros; aprendizaje por refuerzo

1. INTRODUCTION

Combinatorial optimization problems (COPs) are important to many applications in operations research, computer science, and engineering including resource allocation, scheduling, and logistics. Among these applications the mTSP (Mzili et al., 2023) which is challenging due to its NP-hard nature and practical applicability that extending the classical Traveling Salesman Problem (TSP) (Pop et al., 2023) to include multiple salesmen and additional constraints. The mTSP requires effective utilization of multiple salesmen to visit a set of cities ensuring each city is visited exactly once and minimizing total travel distance. This involves handling constraints such as maximum travel distances or costs for specific city assignments and balanced workloads.

Despite many studies of the mTSP and its variants few have explored enhancing MHs with ML techniques. Recent works such as (He et al., 2024) introduced a Multi-Armed Bandit-driven Iterated Local Search (MILS) algorithm improving solution quality but requiring careful parameter tuning. Other approaches like (Belhor et al., 2023) and (Ramanathan et al., 2023) combine Genetic Algorithms (GA) with learning techniques to enhance optimization but adding a degree of complexity. Other methods by (Kusumahardhini et al., 2020) and (Latah, 2016) integrated clustering methods and Ant Colony Optimization (ACO) to simplify the problem but faced the increasing in computational complexity and having sensitivity to parameter tuning.

Moreover, a study by (Nand et al., 2024) introduces a discrete Firefly algorithm (dFA) that balancing exploration and exploitation but depending on experiments to tune the parameters. The work of (Hamza et al., 2023) enhance the Bees Algorithm (BA-2-opt) with new local search methods achieving efficient mTSP solutions but lacking detailed parameter configurations. Additionally, the work of (de Castro Pereira et al., 2023) developed ACO-BmTSP to equalize tour lengths and minimize costs but it has parameter tuning sensitivity. The work of (Gulcu & Ornek, 2019) made adaptive particle swarm optimization (APSO) and hybrid APSO (HAPSO) outperforming other algorithms but requiring careful parameter tuning.

These algorithms in general, facing limitations such as balancing exploration and exploitation, parameter tuning, and initializing the population often requiring extensive experimentation to determine best possible configurations. The quality of the initial population has its effect on the performance of these algorithms potentially reducing the time required for optimization efforts.

This work proposes the DHHO algorithm which adapt the original HHO algorithm (Heidari et al., 2019) to the discrete nature of the mTSP. The DHHO utilize the 3-opt operator (Sui et al., 2021) and Levy flight (Liu & Cao, 2020) to simulate the exploration and exploitation mechanisms of the original algorithm. These modifications including integrate Multiagent Reinforcement Learning (MARL) (Hildebrandt et al., 2023) for initial solution generation and the State-Action-Reward-State-Action (SARSA) (Guo et al., 2023) algorithm for dynamic parameter tuning. This approach allows DHHO to balance exploration and exploitation in the solution search space while utilizing learning mechanism that adapts to problem complexities.

The contributions of this work include the introduction of the DHHO algorithm which integrates ML techniques for enhancing optimization efficiency and solution quality. The DHHO outperforms best known solutions and traditional methods through various mTSP benchmarks effectively balancing exploration and exploitation. The results emphasize the algorithm's capability to consistently achieve better solutions compared to the best-known solutions validating the approach of combining MHs with ML for complex optimization challenges.

2. MATERIALS AND METHODS

2.1. Problem definition

The mTSP is an extension of the classic TSP. While the TSP seeks the shortest possible route for a salesman to visit each city exactly once and return to the origin city, the mTSP involves multiple salesmen as in figure 1, all starting from a single depot, visiting a set of cities, each city has an ID, and then returning to the depot (Cheikhrouhou & Khoufi, 2021). The objective is often to minimize the total distance travelled by all salesmen, although other objectives can also be considered.

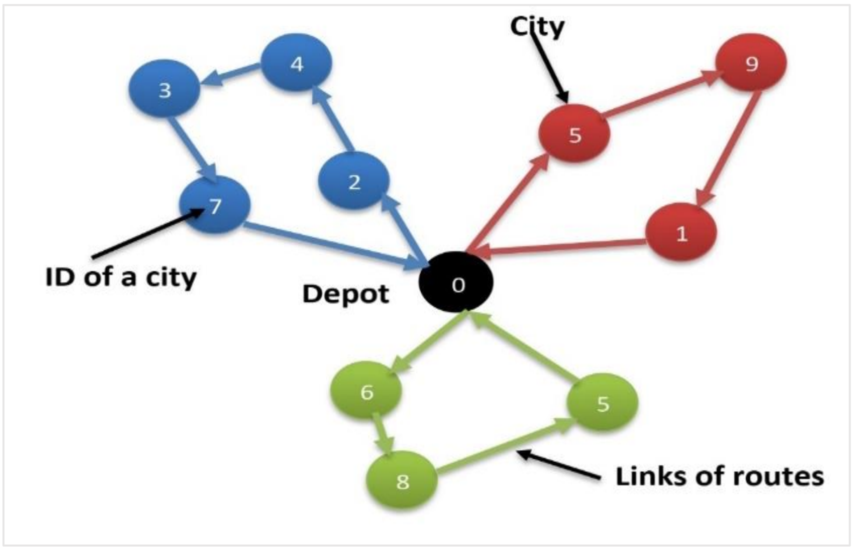


Figure 1. *Representation of mTSP*

The mTSP has many applications, including logistics, delivery services, and scheduling (Singh, 2016). It has several constraints including:

- **Constraint 1:** Refer to minimize the total distance of all salesmen.
- **Constraint 2:** Every city in the solution must visited exactly once by only one salesman except for the depot which is the starting and ending point for all salesmen.
- **Constraint 3:** Salesmen must start and the route at the depot.
- **Constraint 4:** The distance between two cities including the depot must be non-negative.
- **Constraint 5:** The number of salesmen is part of the problem and determined before the tours of each salesman start so that each salesman is responsible of visiting a subset of the cities.
- **Constraint 6:** The graph that represents the cities and the routes links them must be connected which ensures that every city can be attained from the depot by each salesman.

For further understanding of the mathematical formulation and different constraints of the mTSP readers can refer to (Cheikhrouhou & Khoufi, 2021).

2.2. Harris Hawk optimizer

The HHO algorithm is nature inspired optimization algorithm based on the cooperative hunting of Harris hawks known as surprise pounce. This behavior including circling above the prey, approaching from different directions, and perform the surprise pounce when the prey least expecting it. The HHO algorithm simulates this by a mathematical model applying it to solve optimization problems (Heidari et al., 2019). The steps of HHO are the following:

- **Exploration phase:** The algorithm search for a prey (potential solutions) randomly in this step. The position of the hawks is updated using the following equation:

$$X(t+1) = X_{rand} - r_1 \left| X_{rand} - 2r_2 \, X(t) \right| \qquad (6)$$

where $X(t+1)$ is the position vector of in the next iteration (t+1), X_{rand} is the position vector of randomly selected hawk from the current population, r_1 and r_2 are random numbers in the range [0, 1], and $X(t)$ is the current position of the hawk.

- **Transition from exploration to exploitation:**

The transition between exploration and exploitation is governed by the prey's escaping energy which decreases over time. This is represented by the variable E which updated as the following:

$$E = 2\, E_0\, (1 - \frac{t}{T}) \qquad (7)$$

where E_0 is the initial energy of the prey which chosen randomly between range of [-1, 1], t is the current iteration, and T is the maximum number of iterations. The algorithm decides to employ exploration or exploitation strategies based on the value of E.

- **Exploitation phase:**

The exploitation phase is represented using two approaches based on the value of E:

Soft Besiege ($when\ |E| \geq 1\ and\ r$):

$$X(t+1) = \Delta X(t) - E.\,|J.\,X_{rabbit}(t) - X(t)| \qquad (8)$$

Hard Besiege ($when - 1 < E\ <\ 1$):

$$X(t+1) = X(t) - E.\,|J.\,X_{rabbit}(t) - X(t)| \qquad (9)$$

where $X_{rabbit}(t)$ is the position vector of the rabbit (best solution found so far), $\Delta X(t)$ is the difference between the position vector of rabbit and current hawk, and J is a random jump of the rabbit simulates the escape attempt.

Soft Besiege with progressive rapid dives (PRD) and Hard Besiege with PRD are more complex strategies involving the mixture of soft and hard besiege techniques combined with Lévy flight-based dive towards the prey described by:

$$X(t+1) = X_{new}\ if\ q < 0.5 \qquad (10)$$

$$X(t+1) = X_{rabbit}(t) - |X_{rabbit}(t) - X(t)|.\ln\left(\frac{1}{u}\right) if\ q \geq 0.5 \qquad (11)$$

where X_{new} is acquired by equations (8) and (9), and q, u are random number between the range of [0, 1]. These equations allow the HHO to mimic the hunting behavior of the hawks by exploring and exploiting the search space to allow finding best reasonable solutions (Heidari et al., 2019).

2.3. DHHO for mTSP

The application of the proposed algorithm is presented in this section. The initialization of DHHO performed using MARL presented firstly. Then the concepts of 3-opt operator and Levy flight are presented and how they are utilizes in HHO to form discrete version DHHO to optimize the initial solutions of the problem. Lastly the parameters of DHHO are tuned by SARSA dynamically.

Initialization

This section presents the application of a SARSA algorithm within MARL framework to initialize solutions for the DHHO algorithm, targeting the mTSP. The proposed initialization process employs MARL, where each agent, guided by the SARSA algorithm, independently constructs a solution for the mTSP. The SARSA algorithm is a model-free, on-policy RL method, iteratively updates its policy based on the actions taken and the consequent rewards observed, aiming to minimize the total tour length (Zhang et al., 2021). The flow process of initialization the solution in Algorithm 1 is as follow:

Algorithm 1: MARL

```
Inputs: Number of agents ($Na$), Number of cities ($Nc$), Number of episodes (Ne), Learning rate ($\alpha$),
Discount factor ($\gamma$), Exploration probability ($\varepsilon$)
Outputs: Initial solutions for DHHO
Initialize distance matrix Dist_matrix for the problem
Initialize Q-tables $Q_i$ ($i$ = 1, 2, ..., Na) randomly for each agent
For episode = 1 to Ne do:
 For each agent $i$ = 1 to Na do:
  Initialize the starting city (state) $s$
  Initialize the tour as empty list tour
  While not all cities are visited do:
   With probability epsilon select a random action (next city) $a$
    Otherwise select action $a$ with highest Q-value $Q_i$(s, a)
    Execute action $a$ (visit the next city), observe reward $r$ (negative distance)
    Append the city to the tour list
    Observe the new state s′ (next city)
    With probability epsilon select a random action a′
    Otherwise select action a' with highest Q-value $Q_i$(s′, a′)
    Update Q-value $Q_i$ (s, a) using the SARSA update rule from equation 12
    Set state s = s' and action a = a'
   Close the loop when all cities are visited and return to starting city
  If convergence criteria met then break
 Collect the final tours from all agents as initial solutions for DHHO
Return the initial solutions for DHHO
```

- **Initialization**: Each agent starts with randomly initialized Q-table which maps state-action pairs to expected rewards. The initial state is the agent's starting city with the action space comprising the choice of the next city to visit.

- **Policy Selection**: Employ an ε -greedy policy for action selection, allowing agents to explore the action space (choosing the next city) with a probability ε and exploiting known rewarding actions with a probability 1- ε.

- **SARSA Update Rule**: After selecting an action (visiting the next city) and observing the immediate reward (negative of the distance traveled), agents update their Q-values based on the SARSA formula:

  $$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \quad (12)$$

  where $Q(s,a)$ – the current Q-value for the state-action pair, a is the learning rate, r is the immediate reward, γ is the discount factor, and $s'$and a' are the next state and action respectively.

- **Tour Construction**: Agents iteratively select cities based on their policy, updating their Q-table after each move until a complete tour is constructed. The process repeats for a predefined number of episodes or until convergence.

3-Opt operator and levy flight

As stated before, HHO originally designed to meet the requirements of continuous problems. Since the problem tackled in this work has discrete nature, discretization of HHO is required. This is achieved by employing LF and 3-opt operator to mimic the exploration and exploitation criteria in the original design of HHO. To understand the process of DHHO, the initial solution obtained from MARL subject to DHHO which means further optimization process to be applied to the solutions. Before presenting the process of

DHHO, 3-opt and LF need to be comprehended. The exploration phase in DHHO is achieved through the adoption of Lévy Flight (LF) , as shown in Figure 2.

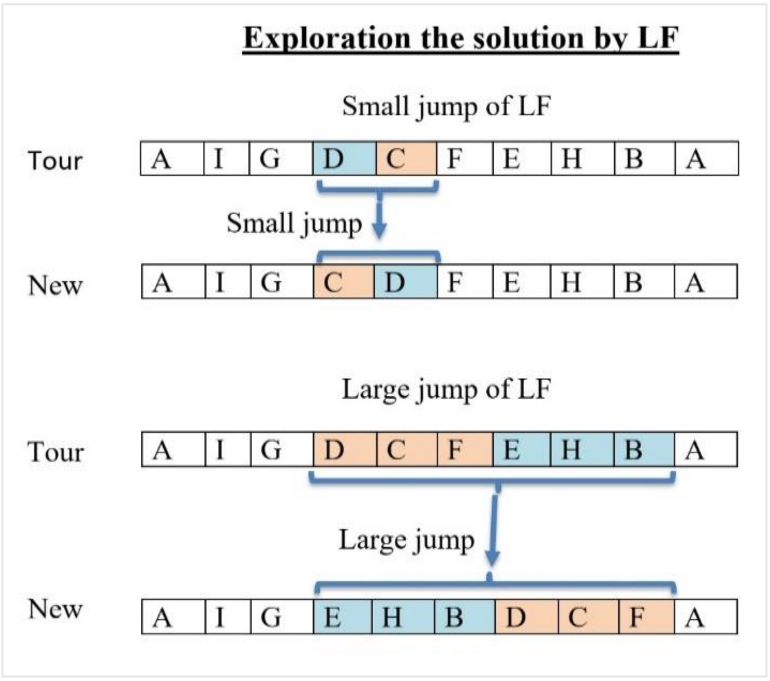


Figure 2. *Representation of exploration process applied on the solutions*

This choice is inspired by the foraging patterns of many animals, which exhibit a tendency for executing long jumps along with of shorter jumps. The advantage of integrating LF within the exploration phase of DHHO lies in its dual ability to utilize extensive search across the global areas of solutions while enabling potential navigation of local neighborhood.

In practical terms, the application of LF in exploring mTSP solutions involves dynamically alternating between large jumps and small jumps within the sequence of city tours. For instance, considering an initial tour configuration such as A-I-G-D-C-F-E-H-B-A as in Figure 2, LF may set a minor adjustment "a small jump" resulting in a small but potentially useful adjustment like swapping adjacent cities. On the other hand, a large jump presented by LF might transform the tour to a sequence such as A-I-G-E-H-B-D-C-F-A, therefore send the search process into potential unexplored area of the search space. The process of LF is shown in Algorithm 2.

Algorithm 2: Levy Flight

```
Inputs: Current tour (Tour), Step size (E), Distance matrix (Dist_matrix)
Outputs: New tour after LF adjustment (New_Tour)
Generate a random number q in the range [0, 1]
if q < 0.5 then:
Select two adjacent cities in the tour                    // Small jump
Swap the selected cities
else:
Select two cities in the tour with a significant distance apart    // Large jump
Move the selected cities to new positions in the tour
Evaluate the new tour length
if new tour length is better than the current tour length then:
Accept the new tour as New_Tour
else:
Retain the current tour as New_Tour
Return New_Tour
```

In the exploitation DHHO, the 3-Opt operator utilized to carefully refine the tours identified during the exploration phase. The 3-Opt operator known for its effectiveness in enhancing solution quality by detaching three non-adjacent edges of the tour and finding the possible reconnections of the resulting segments. This process is designed to find a potential shorter paths that contribute to the minimization of the total travel distance.

An application of the 3-Opt method illustrated in Figure 3 on tour of A-I-G-D-C-F-E-H-B-A involves the selection and removal of three chosen edges (breaking points), for instance (I-G), (D-C), and (E-H), thereby partitioning the tour into four segments.

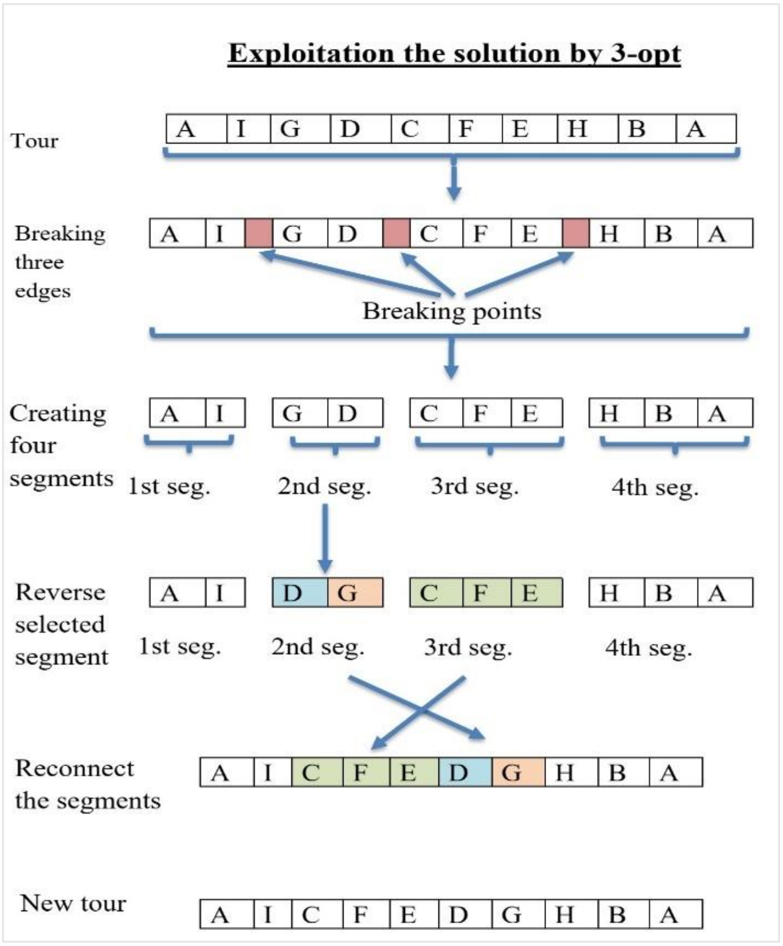


Figure 3. *The representation process of 3-opt operator applied on the solutions*

The reconnection of these segments potentially after reversing or swapping some with the aim to discover a configuration that gives a new tour (solution) and this solution is subject to evaluation to determine is it better or not. A possible outcome of this process might observable as a new tour sequence, A-I-C-F-E-D-G-H-B-A, showing the 3-Opt operator abilities in finding an optimized tour from a given configuration. The stopping criteria of this process is the maximum number of iterations of the algorithm which determined in the next section. The process of 3-opt operator is shown in the Algorithm 3.

Algorithm 3: 3-Opt operator

```
Inputs: Current tour ($Tour$), Distance matrix (Dist_matrix)
Outputs: New tour ($newTour$)
Initialize best distance as the distance of the current $Tour$
Set New_Tour as a copy of the current $Tour$
for each combination of three edges $(i, j, k)$ in the $Tour$ do:
Remove edges $tour_i$-$tour_{i+1}$, $tour_j$-$tour_{j+1}$, $tour_k$-$tour_{k+1}$// Remove three edges
for each possible reconnection of the segments do:        // Reconnect segments
```

```
Reconnect the segments to form a new tour candidate
Calculate the distance of the new tour candidate
if the new tour candidate has a shorter distance than the best distance then:
Update best distance with the distance of the new tour candidate
Set $newTour$ to the new tour candidate
Return $newTour$
```

Discretization of HHO

Adapting HHO in discrete optimization problems such as mTSP involve modifications to maintain its problem-solving abilities. The DHHO achieves this task through the integration of Lévy Flight (LF) for exploration (Liu & Cao, 2020), and utilizing 3-opt for exploitation (Sui et al., 2021). The exploration phase in DHHO moves from random search methods in the original HHO to LF to generate new solutions. By utilizing LF's characteristic long-tail distribution to facilitate both local and global search capabilities. The original random exploration equation (6) is thus replaced by a LF-driven approach:

$$X(t+1) = LF\ (X_{(t)}, E) \tag{13}$$

Where $X(t+1)$ is the position vector of the hawk in the current iteration. LF is the levy flight operation applied on the solution and E is the step size that determine the type of jump and calculated based equation (7). To enhance the exploitation phase, DHHO adopts the 3-opt search operator. This choice reflects the need for a more thorough investigation of the neighborhood near the high-quality solutions allowing for more refinement of tours in the mTSP. The soft and hard besiege strategies are kept conceptually but are adapted to discrete operations facilitated by 3-opt movements by focusing on rearranging tour segments to find shorter routes. Therefore, the equations outlining soft and hard besiege (equations (8) and (9)) are redefined to incorporate 3-opt operations ensuring compatibility with the discrete nature of mTSP solutions as follow:

$$X(t+1) = 3opt\ (X_{(t)}, E) \tag{14}$$

Certain parameters and equations specific to continuous optimization are either modified or omitted. The random jump strength of the rabbit (J), q; and the direct application of continuous equations for position updates are replaced by discrete equivalents that line up with LF and 3-opt strategies. However, r is kept intact to determine whether to apply soft besiege, hard besiege, soft besiege with PRD , and hard besiege with RBP. These adjustments ensure that all aspects of original HHO are fully align with the discrete version of DHHO. The pseudocode of the proposed DHHO is reported in algorithm 4.

Algorithm 4: Discrete Harris Hawk Optimizer

```
Inputs: Population size (number of hawks)(N), Maximum number of iterations (T)
Outputs: The location of the rabbit (best solution) ($X_{rabbit)}$, The fitness value ($Fit_{rabbit}$)
Initialize random population $X_i$ (i = 1, 2, ..., N)
Initialize distance matrix $Dist_{matrix}$ for the problem
while (stopping condition is not met) do:
  Calculate the fitness values of hawks
  Set $X_{rabbit}$ as the location of the rabbit (best location found so far)
  for each hawk $X_i$ do:
    Set initial energy $E_0$ and jump strength $J$
    Update E using Equation (7)
    if (abs(E) >= 1) then:
      Update location vector using $LF$ Equation (12)  // Exploration phase
    if (abs(E) < 1) then:                 // Exploitation phase
      if (r >= 0.5 and abs(E) >= 0.5) then:
        Update location vector using Equation (8)    // Soft besiege
         Apply Equation (13) to increase exploitation
      else if (r >= 0.5 and abs(E) < 0.5) then:
```

```
    Update location vector using Equation (9)    // Hard besiege
     Apply Equation (13) to increase exploitation
   else if (r < 0.5 and abs(E) >= 0.5) then:
    Update location vector using Equation (10)   // Soft besiege with PRD
   else if (r < 0.5 and abs(E) < 0.5) then:
    Update location vector using Equation (11)   // Hard besiege with PRD
  Update $X_{rabbit}$ if the new position of hawk is better
Return $X_{rabbit}$, $Fit_{rabbit}$
```

Parameter tuning of DHHO

To optimize the DHHO algorithm's parameters for solving the mTSP through the SARSA algorithm, the focus on three critical parameters: iterations, population size, and prey energy. These parameters significantly influence the algorithm's performance by controlling the exploration and exploitation balance and the quality of solutions generated for the mTSP (Hussein et al., 2023).

Initially, random values for iterations (I), population size (P), and prey energy (E) are selected within their respective ranges. The Q-table is then established for state-action pairs. States are defined by the values of I, P, and E while actions represented as adjustments to these parameters. The SARSA algorithm iteratively updates the Q-values based on the performance feedback from executing the DHHO algorithm with these parameters.

For example, starting with initial values I=100, P=10, and E=0.5 an action might be chosen to increase the population size (P) to 15. If this change leads to a reduction in the total distance of the TSP tour, the obtained reward is indicating an improvement in solution quality. The Q-value for this state-action pair is updated accordingly, reinforcing the action's effectiveness. In subsequent episodes, actions to adjust the parameters, such as increasing I or modifying E are selected based on the policy determined by the Q-table. Each action measured by the tour distance informs the SARSA algorithm which updates the Q-values to incorporate a new knowledge about the parameter settings effectiveness.

Algorithm 5 details the flow process of parameter tuning.

Algorithm 5: State-Action-Reward-State-Action

```
Inputs: Number of episodes (Ne), Learning rate ($\alpha$), Discount factor ($\gamma$), Exploration probability ($\varepsilon$)
Outputs: Optimized parameters for DHHO ($I, P, E$)

Initialize random initial values for iterations ($I$), population size ($P$), and prey energy ($E$)
Initialize Q-table $Q(s, a)$ for state-action pairs

Define states as combinations of $I, P, and\ E$
Define actions as adjustments to $I, P, and\ E$
For episode = 1 to Ne do:
  Initialize state ($I, P, E$)
  while not stopping condition do:
    With probability $\varepsilon$ select a random action        //adjustment to I, P, or E
    Otherwise select the action with the highest Q-value for the current state
    Execute the DHHO algorithm with the current parameters ($I, P, E$)
    Observe the performance                     //total tour distance
    Calculate the reward $r$
    Observe the new state ($I', P', E'$) after taking the action
    With probability $\varepsilon$ select a random action $a'$
    Otherwise, select the action $a'$ with the highest Q-value for the new state
    Update Q-value $Q(s, a)$ using the SARSA update rule form equation (12)
    Set state $s = s'$ and action $a = a'$
    If convergence criteria met then break
Return the optimized parameters ($I, P, E$) from the Q-table
```

2.4. Experimental setup

The computational experiments were conducted on a laptop equipped with an AMD Ryzen 7 5800H processor, 16GB of RAM, running a Windows 10 operating system. The choice of hardware is significant, as the computational power and memory directly impact the performance of the optimization algorithms, particularly during the intensive iterative processes involved in solving the mTSP. The algorithms were implemented in Python 3.8, utilizing the PyCharm integrated development environment (IDE) for code development and debugging.

The datasets used in this study were sourced from the TSPLIB, a well-known repository for benchmark instances of the Traveling Salesman Problem (TSP) and its variants. Specifically, instances such as Att48, Berlin52, and Bier127 were selected due to their varying complexities and city counts, providing a comprehensive evaluation of the proposed Discrete Harris Hawks Optimization (DHHO) algorithm.

3. RESULTS AND DISCUSSIONS

The computational results obtained from the application of the DHHO algorithm present a significant advancement in solving the mTSP, alongside its non-learning variant of DHHO (NLDHHO) at which its initial solution generated randomly and parameters tuned in offline tuning (Ghani et al., 2004). Table 1 presents the parameters values of the experiment conducted to find the best configuration for the DHHO's parameters number of iterations (I), number of populations (P), and prey energy (P). For the NLDHHO the parameters selected by utilizing Taguchi method (Taguchi & Phadke, 1984).It is worth mentioning that those parameters values selected for each instance regardless the number of salesmen to ensure well tuning of algorithms' behavior. When compared against BKS of dFA (Nand et al., 2024), HAPSO (Gulcu & Ornek, 2019), and BA-2-opt (Hamza et al., 2023), both versions of DHHO show interesting performance emphasize the effectiveness of learning-based parameter optimization and initialization methods in solving combinatorial problems regarding the total distance achieved. In the evaluation through various mTSP instances in table 2 including Att48, Berlin52, Bier127, Pr76, and Rat99 from TSPLIB (Reinelt, 1991), the number in each instance of them indicates the number of cities (e.g., in Att48, there are 48 cities), with each city represented by ID and its coordinates.

Table 1.
Parameter values for each instance

Salesmen	Instances	DHHO parameters			NLDHHO parameters		
		I	P	E	I	P	E
2	Att48	202	33	0.57	266	56	0.5
	Berlin52	373	12	0.49	448	44	0.46
	Bier127	370	31	0.36	487	87	0.93
	Pr76	188	62	0.65	206	90	0.75
	Rat99	171	11	0.23	415	45	0.39
3	Att48	113	59	0.36	288	72	0.61
	Berlin52	120	13	0.43	341	39	0.57
	Bier127	202	47	0.51	364	51	0.97
	Pr76	221	21	0.81	445	38	0.86
	Rat99	152	73	0.28	314	63	0.77
4	Att48	430	13	0.56	485	59	0.59
	Berlin52	187	30	0.63	439	63	0.63
	Bier127	191	42	0.14	472	72	0.97
	Pr76	199	27	0.65	466	85	0.65
	Rat99	363	67	0.25	459	99	0.35

For configurations involving 2, 3, and 4 salesmen the results present a clear difference in solution quality, particularly emphasizing the minimized total distance achieved by the learning variant of DHHO. The DHHO algorithm outperforms NLDHHO and the other algorithms in scenarios with a different number of salesmen with 12 out 15 instances, demonstrating its higher abilities in managing increased complexity and solution space dimensions.

Table 2.
The results of the proposed algorithm against BKS

Salesmen	Instances	BKS	dFA	HAPSO	BA-2-opt	NLDHHO	DHHO
2	Att48	**35047.1**	**35047.1**	37690.9	35380.9	36187.41	**34438.3**
	Berlin52	**8058.3**	**8058.3**	8268.4	8208.1	8139.18	**7961.64**
	Bier127	**124622.4**	**124622.4**	127089.9	130419.9	124916.73	**124047.97**
	Pr76	**115752.8**	116963.4	123341.7	**115752.8**	115924.71	**115113.92**
	Rat99	**1369.1**	**1369.1**	1442.3	1372.5	1438.23	**1327.80**
3	Att48	**38296.3**	38421.8	43845.9	**38296.3**	38824.46	38435.63
	Berlin52	**8545.8**	**8545.8**	9180.5	8594.3	8831.72	**8257.67**
	Bier127	**126806.7**	**126806.7**	137325.9	134036.7	139249.58	**123073.14**
	Pr76	**123239.4**	134889.80	143102.8	**123239.4**	129841.22	124964.75
	Rat99	**1431.5**	1561.54	1734.1	**1431.5**	1519.31	**1363.28**
4	Att48	**42169.88**	43391.14	52716.6	42847.8	43949.81	**41558.24**
	Berlin52	**8820.24**	9047.91	10365	9107.9	9135.39	**8546.82**
	Bier127	**132823.56**	**132823.56**	147184.6	138575.6	133215.72	**132495.48**
	Pr76	**130913.5**	152747.51	168679.15	**130913.5**	131456.40	130997.12
	Rat99	**1522.9**	1801.94	2033.8	**1522.9**	1672.86	**1482.19**

For instance, in the Berlin52 instance with 2, 3, and 4 salesmen DHHO achieved 7961.64, 8257.67, and 8546.82 respectively indicating performance better than the benchmark set by traditional algorithms and also shows a well improvement over the BKS. This the same case in instances Att48, Bier127 , and Rat99 and for 2, 3, and 4 salesmen. That is become clearer in more complex instance "Bier127" where the 124047.97, 123073.14 , and 132495.48 obtained in 2, 3 , and 4 salesmen respectively which is better than other algorithms in the term of less total distance as opposed to NLDHHO which performed poorly in compared to DHHO. This is attributed to the SARSA's role in fine tuning the DHHO parameters, advancing more adaptive exploration and exploitation balance in the search space.

Moreover, the comparative analysis against BKS reveals that DHHO learning-driven approach substantially enhances the algorithm's efficiency particularly evident in instances were traditional heuristics struggle to find better results. The utilization of SARSA for parameter initialization and tuning within DHHO has proven to be effective for its abilities enabling dynamic adjustments that are essential for addressing the complexities of the mTSP effectively. The experimental evidence presented in the study shows the value of integrating RL techniques with MHs optimization algorithms for combinatorial optimization problems. The DHHO algorithm with its learning-based enhancements competes with traditional algorithms as well clear the way for new solution in mTSP.

3.1. Performance analysis

In this study, the performance of the DHHO algorithm evaluated across various mTSP instances with different numbers of salesmen. In table 3, the standard deviation (SD) and SD/Mean ratio were used to assess the consistency of the results, where lower values indicate higher consistency. These metrics is a reflection of 30 runs to test the algorithms' consistency. For instance, "Att48" in 2 salesmen exhibited a low SD of 38.27 and an SD/Mean ratio of 0.11%, demonstrating high reliability. Similarly, "Berlin52" showed a consistent performance with an SD of 15.3and an SD/Mean ratio of 0.19%. The mean objective function values provided a measure of central tendency, with lower means reflecting better performance. "Att48" had a mean value of 34482.48, indicating strong overall performance, while “Berlin52” had a mean value of 7990.12, also suggesting effective optimization.

Computational efficiency was assessed through the minimum and average computational times. Table 3 reveals that “Att48” in 2 salesmen achieved its best result in a minimum time of 0.94 seconds, with an average computational time of 1.88 seconds, indicating efficient computation. “Berlin52” demonstrated similar efficiency with a minimum time of 0.71 seconds and an average time of 1.34 seconds.

However, some instances revealed limitations. For example, “Bier127” in 2 salesmen exhibited a higher SD of 5531.49 and an SD/Mean ratio of 4.45%, indicating greater variability in the results. Additionally the “Pr76” instance in 2 salesmen had mean value of 115205.11 with SD of 135.73 reflecting less consistent performance in compare to other instances. The computational time for “Bier127” instance showed higher variability with minimum time of 2.47 seconds and average time of 3.85 seconds which indicate less efficient computation.

These results highlight the DHHO ability to produce high quality solutions with reasonable time for certain instances such as “Att48” and “Berlin52” instances. However these variability and computational time for instances like “Bier127” and “Pr76” suggest areas for further improvement.

The computational complexity of DHHO algorithm analyzed to comprehend how it is scale with problem size. The algorithm time complexity estimated to be O(n^2) where n represents the number of cities. This indicates the computational time increases quadratically with the problem size which consistent with the results shown in Table 3. The minimum and average time reported in Table 3 reflects the practical run time for problem instances which align with the expected theoretical growth in computational resources as the input size increase.

Table 3.
Descriptive statistical metrics values of DHHO performance

Instance	Salesmen	SD	Mean	SD/Mean ratio	Min Time	Avg Time
Att48	2	38.27	34482.48	0.11 %	0.94	1.88
Berlin52		15.3	7990.12	0.19 %	0.71	1.34
Bier127		5531.49	124174.12	4.45 %	0.93	1.55
Pr76		135.73	115205.11	0.12 %	0.42	1.79
Rat99		18.22	1361.46	1.34 %	0.15	1.7
Att48	3	17.93	38453.59	0.05 %	0.98	1.9
Berlin52		16.55	8284.79	0.2 %	1.76	1.92
Bier127		44.82	123115.92	0.04 %	0.59	1.42
Pr76		19.05	124981.86	0.02 %	4.82	5.33
Rat99		14.76	1385.71	1.07 %	8.88	8.93
Att48	4	29.46	41596.16	0.07 %	2.38	2.80
Berlin52		25.96	8583.25	0.3 %	2.49	7.04
Bier127		37.51	132536.28	0.03 %	5.47	12.41
Pr76		27.36	131028.46	0.02 %	3.84	6.17
Rat99		17.41	1508.84	1.15 %	3.35	5.98

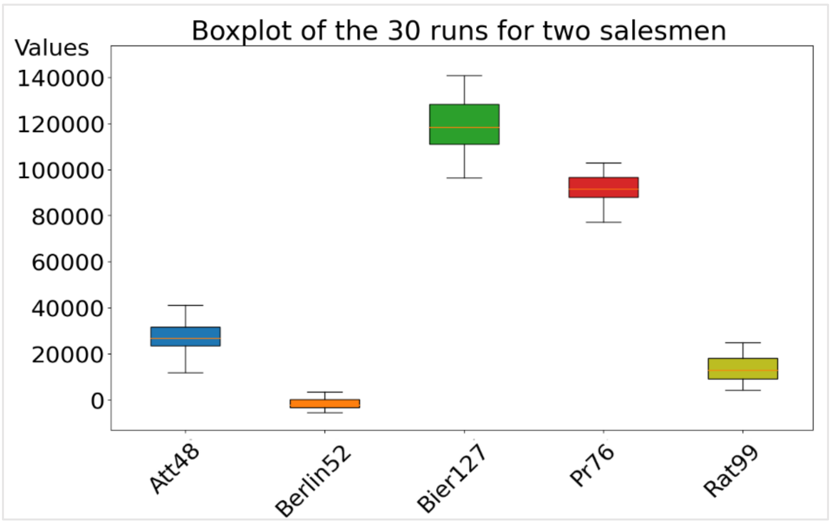
Additionally, The Wilcoxon signed rank test is conducted to compare the performance of the DHHO algorithm against the NLDHHO algorithm across various instances. The statistical test applied to evaluate whether the observed difference in solution quality between DHHO and NLDHHO is statistically significant. The p-values obtained from the Wilcoxon test indicate the level of statistical significance of the performance differences between DHHO and NLDHHO. The p-value less than 0.05 is considered statistically significant in general meaning that there is strong evidence to suggest that the performance difference is not due to random chance. For 2 salesmen all instances show statistically significant differences ($p < 0.05$) with p-values ranging from 0.01862 (Att48) to 0.03725 (Bier127). This suggests that DHHO significantly outperforms NLDHHO in all the tested instances when using 2 salesmen. For 3 salesmen The Att48 instance show a p-value of 0.09672 which is not statistically significant (p > 0.05). However, the other instances (Berlin52, Bier127, Pr76, and Rat99) have p-values below 0.05 indicating significant differences in performance in favor for DHHO. For 4 salesmen all instances demonstrate statistically significant differences (p < 0.05) with the Berlin52 instance showing the most significant difference (p = 0.01873).

Table 4.
Wilcoxon signed ranked test for DHHO against NLDHHO

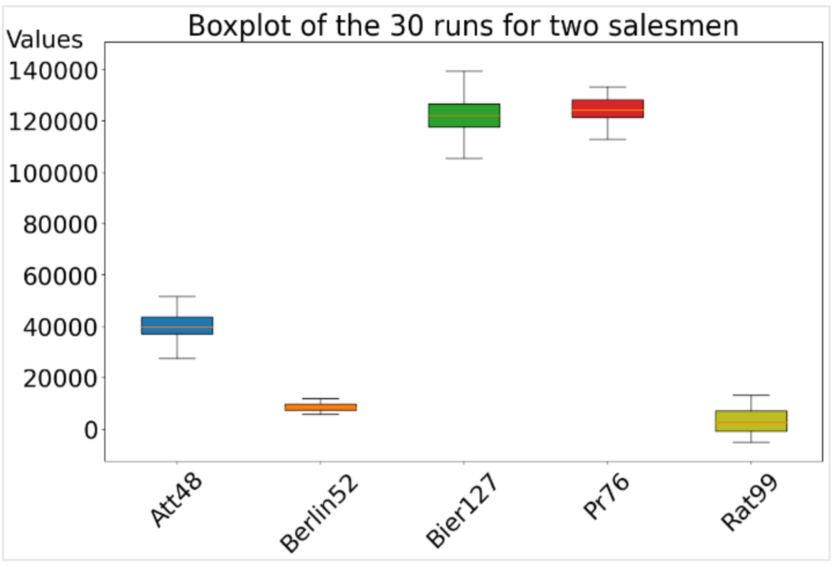
Instance	Salesmen	Wilcoxon test
Att48	2	0.01862
Berlin52		0.01893
Bier127		0.03725
Pr76		0.02707
Rat99		0.03634
Att48	3	0.09672
Berlin52		0.04287
Bier127		0.03865
Pr76		0.03362
Rat99		0.027394
Att48	4	0.04865
Berlin52		0.01873
Bier127		0.02718
Pr76		0.03949
Rat99		0.03452

Furthermore, the performance analysis of the DHHO algorithm applied to solve the mTSP presented focusing on the vision provided by Figure 4. These figures are visualization of the 30 runs of the solution obtained by DHHO displayed as boxplot to test the consistency of the proposed DHHO. A boxplot is a standard way of displaying the distribution of data based on five components: minimum value, first quartile, median, third quartile, and maximum. It can also highlight outliers in the data set if available. This visualization is useful in comparing the spread and skewness of algorithm performance over multiple runs. In the context of DHHO's analysis boxplots encapsulate the variability and central tendency of the total distances achieved by the proposed algorithm which provide a clear comparative perspective. Figure 4 (a) draws the boxplot for instances including two salesmen. The skewness observed in these plots emphasize an orientation towards lower total distances (minimum value) indicating efficient solutions.

(a)



b)



(c)

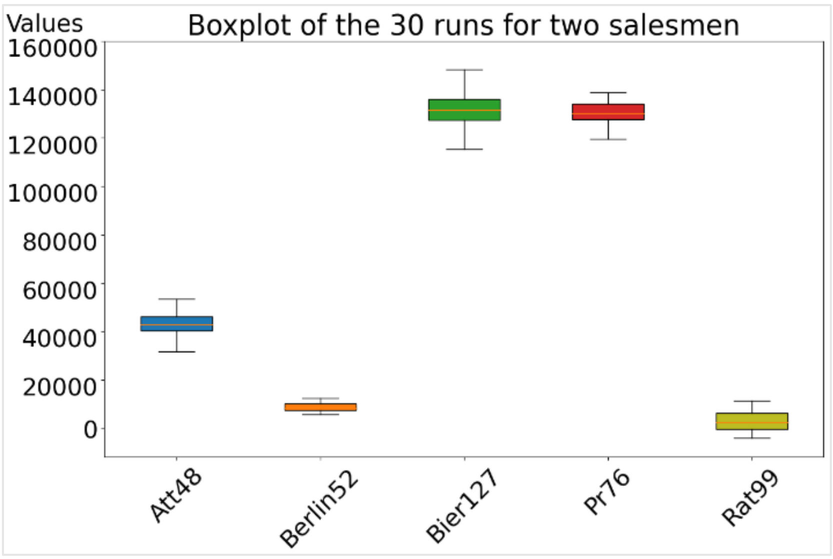


Figure 4. *The boxplot of the results for: (a) two salesmen (b) three salesmen (c) four salesmen*

This skewness of the distribution suggests that DHHO constantly pushes the boundary towards best found solutions as evidenced by the concentration of total distances on the lower end of the box. Similarly, Figure 4 (b) and Figure 4 (c) provide this analysis of mTSP instances with three and four salesmen respectively. The distribution shapes validate DHHO's robustness in handling increased complexity. The learning based DHHO shows a plain skewness towards better performance indicating that the SARSA enhanced parameter tuning effectively navigates the solution space to find better tours. The appearance of these boxplots and their skewness can be attributed to the essential learning capabilities of DHHO. By utilizing SARSA for initialization and parameter tuning DHHO adapts its search strategy dynamically. This adaptability is essential in solving mTSP instances efficiently as it allows the algorithm to balance the exploration and exploitation which lead to better outcomes as compared to traditional or non-learning approaches.

While the DHHO shown promising results in solving the mTSP several limitations should be addressed. The computational complexity may limit its applicability to very large problem instances. Additionally, the experiments were conducted on symmetric mTSP instances so the algorithm did not tested to other problem variations such as asymmetric instances or those with additional constraints. The algorithm's performance is also sensitive to parameter settings requiring extensive tuning. Moreover, the algorithm has not yet been tested in real-world scenarios leaving its practical applicability open. Finally, the scalability of the DHHO algorithm to significantly larger problem sizes remain an open question which could affect its utility in handling more extensive optimization problems.

CONCLUSIONS

In this study the DHHO algorithm presented enhance with SARSA-based dynamic parameter tuning and MARL for initialization of the population to solve the mTSP. The integration of MHs with ML for DHHO aimed to improve solution quality and computational efficiency across various problem instances. The experimental results shown that DHHO outperforms traditional optimization algorithms achieving an average reduction in total distance across 12 out of 15 benchmark instances from TSPLIB. The statistical significance of these improvements confirmed through the Wilcoxon signed rank test with p-values below 0.05 confirms the robustness and effectiveness of the proposed algorithm. Despite these promising outcomes the study also identified certain limitations. The quadratic time complexity of the DHHO algorithm may impact its scalability in very large problem instances. Additionally, the generalizability of the algorithm to different variations of the mTSP such as asymmetric instances remains an open area for further research. Future work should focus on optimizing the computational complexity of the DHHO algorithm to enhance its scalability and applicability to larger problem instances. Exploring the integration of other ML techniques and further testing the algorithm on real-world instances could provide more understanding into its potential. The findings of this study contribute to the growing body of research on hybrid optimization algorithms highlighting the value of combining MHs with ML to solve complex combinatorial optimization problems.

FINANCING

The authors did not receive sponsorship to carry out this study-article.

CONFLICT OF INTEREST

There is no conflict of interest related to the subject matter of the work.

AUTHORSHIP CONTRIBUTION

Conceptualization, data curation, formal analysis, acquisition of funds, research, methodology, project administration, resources, software, supervision, validation, visualization, writing - original draft, writing - review and editing, achieved: Hussein, A. A., Hameed, M. A. & Ahmed, S. H.

REFERENCES

Belhor, M., El-Amraoui, A., Jemai, A., & Delmotte, F. (2023). Learning-Based Metaheuristic Approach for Home Healthcare Optimization Problem. *Comput. Syst. Sci. Eng.*, *45*(1), 1–19. https://doi.org/10.32604/csse.2023.029058

Cheikhrouhou, O., & Khoufi, I. (2021). A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy. *Computer Science Review*, *40*, 100369. https://doi.org/10.1016/j.cosrev.2021.100369

de Castro Pereira, S., Solteiro Pires, E. J., & de Moura Oliveira, P. B. (2023). Ant-Balanced Multiple Traveling Salesmen: ACO-BmTSP. *Algorithms*, *16*(1), 37. https://doi.org/10.3390/a16010037

Ghani, J. A., Choudhury, I. A., & Hassan, H. H. (2004). Application of Taguchi method in the optimization of end milling parameters. *Journal of Materials Processing Technology*, *145*(1), 84–92. https://doi.org/10.1016/S0924-0136(03)00865-3

Gulcu, S. D., & Ornek, H. K. (2019). Solution of multiple travelling salesman problem using particle swarm optimization based algorithms. *International Journal of Intelligent Systems and Applications in Engineering*, *7*(2), 72–82. https://doi.org/10.18201//ijisae.2019252784

Guo, Y., Wang, Y., Yang, I.-H., & Sycara, K. (2023). Reinforcement learning methods for network-based transfer parameter selection. *Intelligence & Robotics*, *3*(3), 402–419. https://doi.org/10.20517/ir.2023.23

Hamza, A., Darwish, A. H., & Rihawi, O. (2023). A new local search for the bees algorithm to optimize multiple traveling salesman problem. *Intelligent Systems with Applications*, 200242. https://doi.org/10.1016/j.iswa.2023.200242

He, P., Hao, J.-K., & Xia, J. (2024). Learning-guided iterated local search for the minmax multiple traveling salesman problem. *ArXiv Preprint ArXiv:2403.12389*. https://doi.org/10.48550/arXiv.2403.12389

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems*, *97*, 849–872. https://doi.org/10.1016/j.future.2019.02.028

Hildebrandt, F. D., Thomas, B. W., & Ulmer, M. W. (2023). Opportunities for reinforcement learning in stochastic dynamic vehicle routing. *Computers & Operations Research*, *150*, 106071. https://doi.org/10.1016/j.cor.2022.106071

Hussein, A. A., Yassen, E. T., & Rashid, A. N. (2023). Grey Wolf Optimizer for Green Vehicle Routing Problem. *International Journal of Intelligent Engineering & Systems*, *16*(5). https://doi.org/10.22266/ijies2023.1031.53

Kusumahardhini, N., Hertono, G. F., & Handari, B. D. (2020). Implementation of K-Means and crossover ant colony optimization algorithm on multiple traveling salesman problem. *Journal of Physics: Conference Series*, *1442*(1), 012035. https://doi.org/10.1088/1742-6596/1442/1/012035

Latah, M. (2016). Solving multiple TSP problem by K-means and crossover based modified ACO algorithm. *International Journal of Engineering Research and Technology*, *5*(02). https://doi.org/10.17577/IJERTV5IS020474

Liu, Y., & Cao, B. (2020). A novel ant colony optimization algorithm with Levy flight. *Ieee Access*, *8*, 67205–67213. https://doi.org/10.1109/ACCESS.2020.2985498

Mzili, I., Mzili, T., & Riffi, M. E. (2023). Efficient routing optimization with discrete penguins search algorithm for MTSP. *Decision Making: Applications in Management and Engineering*, *6*(1), 730–743. https://doi.org/10.31181/dmame04092023m

Nand, R., Chaudhary, K., & Sharma, B. (2024). Single Depot Multiple Travelling Salesman Problem Solved With Preference-Based Stepping Ahead Firefly Algorithm. *IEEE Access*, *12*, 26655–26666. https://doi.org/10.1109/ACCESS.2024.3366183

Pop, P. C., Cosma, O., Sabo, C., & Sitar, C. P. (2023). A Comprehensive Survey on the Generalized Traveling Salesman Problem. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2023.07.022

Ramanathan, T., Suresh, S., & Rao, T. S. (2023). Multiple Depot MTSP using Genetic Algorithm and Reinforcement Learning. *2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*, 440–446. https://doi.org/10.1109/ICAISS58487.2023.10250669

Reinelt, G. (1991). TSPLIB—A traveling salesman problem library. *ORSA Journal on Computing*, *3*(4), 376–384. https://doi.org/10.1287/ijoc.3.4.376

Singh, A. (2016). A review on algorithms used to solve multiple travelling salesman problem. *International Research Journal of Engineering and Technology (IRJET)*, *3*(4), 598–603. https://www.irjet.net/archives/V3/i4/IRJET-V3I4120.pdf

Sui, J., Ding, S., Liu, R., Xu, L., & Bu, D. (2021). Learning 3-opt heuristics for traveling salesman problem via deep reinforcement learning. *Asian Conference on Machine Learning*, 1301–1316. https://proceedings.mlr.press/v157/sui21a

Taguchi, G., & Phadke, M. S. (1984). Quality engineering through design optimization. *Quality Control, Robust Design, and the Taguchi Method*, 77–96. https://doi.org/10.1007/978-1-4684-1472-1_5

Zhang, K., Yang, Z., & Başar, T. (2021). Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, 321–384. https://doi.org/10.1007/978-3-030-60990-0_12