Decision Making: Applications in Management and Engineering
Vol. 6, Issue 2, 2023, pp. 150-176.
ISSN: 2560-6018
eISSN: 2620-0104
DOI: https://doi.org/10.31181/dmame622023644

ARTIFICIAL RAT OPTIMIZATION WITH DECISION-MAKING: A BIO-INSPIRED METAHEURISTIC ALGORITHM FOR SOLVING THE TRAVELING SALESMAN PROBLEM

Toufik Mzili [1*], Ilyass Mzili[2], and Mohammed Essaid Riffi [1]

[1] Department of Computer Science, Faculty of Science, Chouaib Doukkali, University, EI Jadida , Morocco
[2] Department of Management, Faculty of Economics and Management, Hassan First University, Settat, Morocco

Received: 4 January 2023;
Accepted: 15 March 2023;
Available online: 11 April 2023.

Original scientific paper

Abstract: *In this paper, we present the Rat Swarm Optimization with Decision Making (HDRSO), a hybrid metaheuristic algorithm inspired by the hunting behavior of rats, for solving the Traveling Salesman Problem (TSP). The TSP is a well-known NP-hard combinatorial optimization problem with important transportation, logistics, and manufacturing systems applications. To improve the search process and avoid getting stuck in local minima, we added a natural mechanism to HDRSO by incorporating crossover and selection operators. In addition, we applied 2-opt and 3-opt heuristics to the best solution found by HDRSO. The performance of HDRSO was evaluated on a set of symmetric instances from the TSPLIB library, and the results demonstrated that HDRSO is a competitive and robust method for solving the TSP, achieving better results than the best-known solutions in some cases.*

Key words: *Bio-inspired; Metaheuristics; Rat Swarm Optimizer (RSO); Combinatorial optimization; TSP; Artificial intelligence (AI); Swarm intelligence (SI); Modeling systems.*

1. Introduction

Optimization, planning, and decision-making in real-time are essential in every aspect of our lives, from daily decision-making to the operations of large companies. However, these decisions can often be complex, with multiple factors and potential drawbacks. By looking at how large companies and mega-companies approach decision-making, we can gain insight into how to make better choices. These companies often face high stakes, with significant potential gains and losses. They use various methods and tools to address these complex optimization problems, which

* Corresponding author.
E-mail addresses: mzili.t@ucd.ac.ma (T. Mzili), saidriffi@gmail.com (M.E. Riffi), dr.mzili.ilyass@gmail.com (I. Mzili)

can be classified based on their computation time and solution quality. Some methods prioritize speed and may not always find the optimal solution, while others prioritize a very high solution quality, which may come at the cost of longer computational time. Ultimately, the performance and efficiency of these methods depend on both their optimality and the time required for implementation.

Combinatorial optimization problems (COPs) are an important area of study within operations research, with applications in various fields such as industry, urban management, biology, and technology (Peres & Castelli, 2021). When studying these problems, it is important to consider factors such as the available time and resources, the potential benefits of the study, and the available tools and computing power. To solve COPs, there are several classes of methods, including exact and deterministic methods (Chung & Freund, 2022). These methods typically involve enumerating the possible solutions in the search space, using techniques such as boundary calculations and heuristics to guide the search and improve efficiency. Traditional methods such as separation and progressive evaluation techniques (SEP) or backtracking algorithms fall under this category. While exact methods can be used to find optimal solutions for problems of moderate size, their computational time tends to increase exponentially with the size of the problem, making them less practical for larger applications.

When the need for an optimal solution is not as pressing, approximate approaches can provide an efficient solution for large optimization problems. These techniques, such as greedy approaches and iterative improvement, have been used by practitioners for many years and have proven effective in various contexts. For example, Lin and Kernighan's approach is widely considered the best algorithm for the traveling salesman problem. These approximate methods can balance computational time and solution quality for certain types of problems.

In recent years, significant progress has been in developing powerful and general approximate methods known as metaheuristics. These methods, which include neighborhood approaches such as simulated annealing and tabu search (Prajapati et al., 2020) and evolutionary algorithms such as genetic algorithms (Sun, 2015) and evolutionary strategies (Slowik & Kwasnicka, 2020), have enabled the development of approximate solutions for large-scale classical optimization problems and previously unmanageable applications (Ezugwu et al., 2021). Metaheuristics have gained increasing attention in operations research and artificial intelligence in recent years.

There are several reasons why metaheuristics have become increasingly popular in recent years:

- They have strategies in place to guide the search for optimal solutions.
- They can efficiently explore the search space to find (near) optimal solutions.
- The techniques that make up metaheuristic approaches range from simple local search algorithms to complex learning processes.
- They have mechanisms to avoid getting stuck in suboptimal regions of the search space.
- They can incorporate problem-specific heuristics into the search process, but a higher-level strategy controls these.
- They can use the experience gained during the search process to guide the remainder of the search better.

Table 1 provides a classification of several types of metaheuristics that can be distinguished.

Table 1. A brief review of metaheuristic algorithms

Type of Metaheuristic	Metaheuristic	Author and year
Algorithms metaheuristics	Particle Swarm Optimization (PSO)	(Kennedy & Eberhart, 1995)
	Firefly Algorithm (FA)	(Xu et al., 2022; Yang, 2009)
	Bat Algorithm (BA)	(Saji & Riffi, 2016; Yang, 2010)
	Salp Swarm Algorithm (SSA)	(S. Mirjalili et al., 2017)
	Wolf Optimization (GWO)	(Medjahed et al., 2016)
	Gorilla Troops Optimizer (GTO)	(Ginidi et al., 2021)
	Grasshopper Optimization Algorithm (GOA)	(S. Z. Mirjalili et al., 2018)
Physics-based algorithms	Simulated annealing (SA)	(Kirkpatrick et al., 1987)
	Lichtenberg Algorithm (LA)	(Pereira et al., 2021)
	Gravitational search algorithm (GSA)	(Rashedi et al., 2009)
	Black hole algorithm (BB)	(Abualigah et al., 2022)
Evolutionary algorithms :	Genetic Algorithm (GA)	(Sun, 2015)
	Genetic Programming (GP)	(Koza & Poli, 2005)
	Evolutionary programming (EP)	(Opara & Arabas, 2019)
	Biogeography Based Optimizer (BBO)	(Simon, 2008)
	Tree-Seed Algorithm (TS)	(Cinar et al., 2020)
Human algorithms	Harmony Search (HS)	(Lee & Geem, 2004)
	Imperialist Competitive Algorithm (ICA)	(Atashpaz-Gargari & Lucas, 2007)
	Tabu Search (TS)	(Barbarosoglu & Ozgur, 1999)
	Heat Exchange Optimization (TEO)	(Kaveh & Dadras, 2017)

The study of optimization and NP-hard problem-solving, including metaheuristics, has been influenced by the behavior of animals in nature (Tanaev et al., 1994). One well-known and extensively studied problem in this field is the traveling salesman problem (TSP) (Mzili et al., 2020), which belongs to the class of NP-hard optimization problems. The TSP involves finding the shortest route that visits a list of cities, passing through each city only once. While the problem may initially seem simple, no known algorithm can quickly find an exact solution for all cases. Furthermore, computational complexity increases exponentially with the number of cities, making it a useful test case for optimization techniques. The TSP has many practical applications, including in astronomy, logistics, transportation, telecommunications, and scheduling. Metaheuristic algorithms have successfully solved the TSP and other similar problems, demonstrating their versatility and effectiveness. These algorithms use search techniques to explore the search space efficiently, often focusing on specific areas of interest.

The contributions of this paper are as follows:

- Presentation of the Rat Swarm Optimizer (RSO), a new robust optimizer inspired by wild rats' attack and hunting behavior, outperforms many known metaheuristics and effectively solves the discrete traveling salesman problem.
- Proposed a hybrid approach using RSO to solve a widely applicable and influential combinatorial problem with potential applications in various domains.
- Developed a uniform crossover and mutation operator mechanism to improve performance in the exploration phase, thereby conserving information throughout the search space and balancing exploration and exploitation.
- Use local Lin-Kernighan searches to increase efficiency and an acceptance and solution search strategy to avoid getting stuck in local optima.
- Introduction and testing of a new random parameter, T, to balance the workload of the auxiliary operators.
- Tested the performance of the proposed algorithm on more than 26 instances of the TSPLIB library and used the parametric student's t-test and the non-parametric Wilcoxon test to compare the proposed algorithm to other models.
- Comparison of the proposed HDRSO algorithm with the baseline algorithm and five recently developed bio-inspired metaheuristics: DJAYA, RNN-SA, GGSC-SSA, DSSA, and DSOS, to demonstrate its superior performance.

This paper is structured as follows: Section 1 introduces the general topic. Section 2 represents some related works; Section 3 discusses the Traveling Salesman Problem. Section 4 presents the Rat Swarm Optimizer. Section 5 proposes an improved and hybrid Rat Swarm Optimizer. Section 6 presents the results and discussion, and finally, the conclusion.

2. Related works

The Traveling Salesman Problem (TSP) is widely studied in optimization, with numerous algorithms being developed to solve it efficiently. The TSP seeks to find the shortest possible route for a salesman who must visit a set of cities exactly once and return to the starting city. The TSP is an NP-hard problem, meaning that its solution time grows exponentially as the size of the problem increases. Nevertheless, a vast literature on the TSP has many approaches to solving it using various algorithms.

Swarm intelligence (SI) algorithms are a class of metaheuristic optimization techniques based on social animals' collective behavior. The intelligent behavior of social insects, such as ants, bees, and termites, inspires these algorithms. Serval swarm intelligence algorithms have been proposed to solve the TSP in recent years. Some of the popular swarm intelligence algorithms that have been used to solve the TSP include Ant Colony Optimization (ACO) (Dorigo & Gambardella, 1997), Particle Swarm Optimization (PSO) (Kennedy & Eberhart, 1995), Artificial Bee Colony (ABC)(Anuar et al., 2016), Firefly Algorithm (FA) (Yang, 2009), Bat Algorithm (BA) (Yang, 2010), Cuckoo Search (CS) (Yang & Deb, 2009), and Wolf Search Algorithm (WSA) (Medjahed et al., 2016).

In addition to the swarm intelligence algorithms, other approaches have been proposed to solve the TSP. These approaches include Genetic Algorithms (GA), Simulated Annealing (SA) (Kirkpatrick et al., 1987), Tabu Search (TS) (Barbarosoglu & Ozgur, 1999), Iterated Local Search (ILS) (Lourenço et al., 2010) Variable Neighborhood Search (VNS)(Mladenović & Hansen, 1997), Memetic Algorithms (MA)(Cotta et al., 2018), and Hybrid Algorithms that combine different techniques.

Genetic Algorithm GA (Sun, 2015) is a popular optimization technique that mimics the process of natural selection. It has been used to solve the TSP, and several variants of GA have been proposed. SA is another optimization technique that simulates the process of annealing in metals. It has been applied to solve the TSP, and several variants of SA have been proposed. TS is another optimization technique that uses short-term memory to avoid visiting the same city twice. It has been applied to solve the TSP, and several variants of TS have been proposed. ILS is another optimization technique that combines local search with perturbation. It has been applied to solve the TSP, and several variants of ILS have been proposed. VNS is another optimization technique that uses a sequence of neighborhoods to explore the search space. It has been applied to solve the TSP, and several variants of VNS have been proposed. Finally, MA is another optimization technique combining local and global searches.

Recent research in the optimization field has focused on applying bio-inspired metaheuristics to solve real-world problems. Osaba et al. (2020) reviewed recent research on TSP and the application of bio-inspired metaheuristics in solving it. A. Kumar et al. (2021) provided a comprehensive overview of metaheuristic optimization techniques and their applications in engineering. A. Kumar et al. (2021) reviewed optimization techniques for petroleum engineering, while Uniyal et al. (2022) provided an overview of nature-inspired metaheuristic algorithms for optimization. A. Kumar et al. (2021) used nature-inspired optimization algorithms to optimize the availability-cost of a butter oil processing system, while Rawat et al. (2022) provided a state-of-the-art survey on the applications of the Analytical Hierarchy Process (AHP). Finally, J. Kumar et al. (2021) discussed the use of Multi-Criteria Decision-Making (MCDM).

3. The traveling salesman problem

3.1. Introducing the traveling salesman problem

The TSP is a classic and old NP-hard problem. The TSP problem consists in finding the smallest path to visit a list of cities by passing through a city only once and returning to the starting city. To help the commercial traveler to save more time. money. and effort.

This problem can be defined mathematically as follows:

C = $\{c_{(1)}. c_{(2)}.... c_{(i)}. ...c_{(n)}\}$ list of n cities (nodes). and a distance (weight) matrix

D = $(d_{(i.j)})_{n\times n}$ where $d_{(i.j)}$ represents the Euclidean distance between the cities $c_{(i)}$ and$c_{(j)}$. The goal of this problem is to find the shortest tour $\pi = (\pi_{(1)}.\pi_{(2)}. ... . \pi_{(n)}) \in$ L (L set of all possible tours) that minimizes the following cost function

$$F(\pi) = Min_{\pi \in L}\left(\sum_{k=1}^{n-1} d\pi_{(k)}\pi_{(k+1)} + \mathrm{d}\pi_{(n)}\pi_{(1)} \right) \quad (1)$$

For a 2-dimensional case, the Euclidean distance between any two cities $c_{(i)}$ and $c_{(j)} \in$ C is determined by:

$$d_{(i.j)} = \sqrt{\left(x_{(i)} - x_{(j)}\right)^2 + \left(y_{(i)} - y_{(j)}\right)^2} \quad (2)$$

Where $\left(x_{(i)}.y_{(i)}\right)$ and $\left(x_{(j)}.y_{(j)}\right)$ are respectively the coordinates of cities $c_{(i)}$ and $c_{(j)}$.

We say that TSP is symmetrical if $d_{(i.j)} = d_{(j.i)} \ \forall i.j \in N$. otherwise, if this equality is not satisfied is for at least a pair $(i.j)$ we call the problem asymmetric TSP (ATSP).

3.2. Importance of solving the TSP

The Traveling Salesman Problem (TSP) is a fundamental problem of finding the shortest possible route to visit a set of locations and return to the starting point. TSP has many applications in logistics, transportation, and manufacturing. Effective solutions to TSP can provide significant cost savings and efficiency improvements in these areas.

In logistics, TSP is essential for determining the most efficient delivery routes for vehicles, which reduces travel time and distance. As a result, this can result in cost savings and improved customer satisfaction through shorter delivery times.

Similarly, in transportation, TSP can help plan optimal routes for public transportation, including buses and trains, thereby reducing fuel consumption and emissions and improving the overall efficiency of the transportation system.

In manufacturing, TSP is used to optimize the order in which tasks are processed on a production line, reducing overall processing time and minimizing machine idle time. These benefits can translate into increased productivity and significant cost savings for companies. Thus, the ability to effectively resolve TSP can profoundly impact decision-making in various areas, leading to improved efficiency and substantial cost savings.

4. Presentation of rat swarm optimizer

The Rat Swarm Optimizer (RSO) is a metaheuristic algorithm that uses the collective behavior of rats in a swarm as a model for optimization. The algorithm is inspired by the hunting and aggressive behavior of rats in the wild, which is used to model the exploration and exploitation phases of the search for solutions to optimization problems. RSO effectively solves various continuous optimization problems and has been used in many different applications. The hunting and aggressive behavior of rats in the wild is the main inspiration for the RSO algorithm, which mathematically models this behavior to optimize solutions to difficult problems. This behavior is characterized by the social intelligence and territoriality of rats and their ability to engage in complex behaviors such as jumping and running.

The two main behaviors that are the basis of the RSO algorithm are:

- Swarm hunting behavior: When rats think they have located their prey, they will designate a captain and follow him, allowing them to cover the entire search area.
- Fighting behavior with prey: To hunt their prey, rats will enter conflict with them. This conflict may result in the death of some rats, which is modeled in the algorithm as the cancellation of a particular solution. These behaviors are used to guide the exploration and exploitation phases of the search for solutions in the RSO algorithm.

Figure 1 shows the movement of rats around the prey in a 2D space.

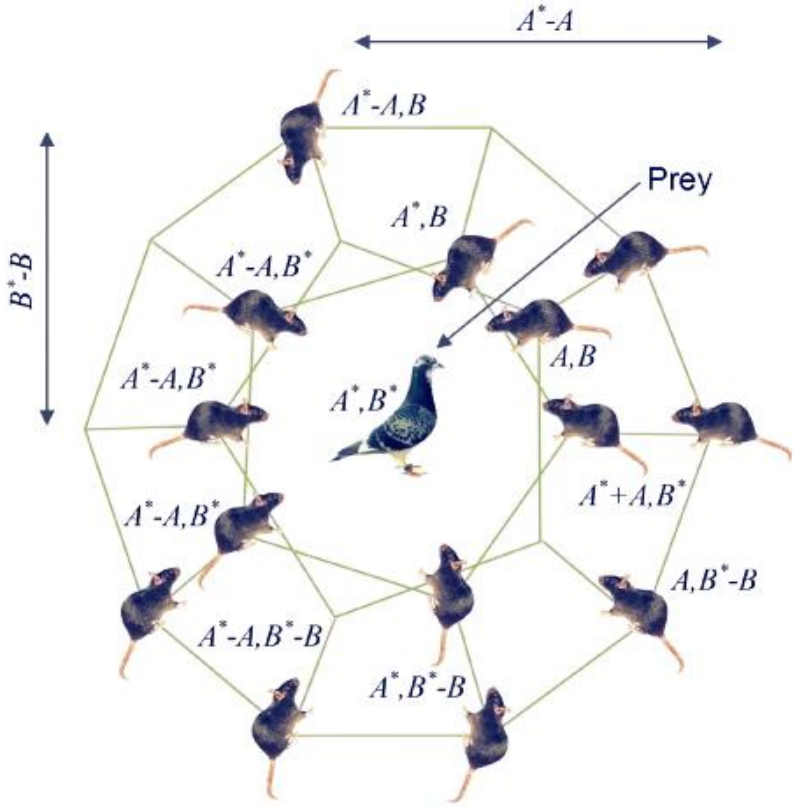


Figure 1. Movement of rats in 2D

4.1. Mathematical modeling of the RSO algorithm

Pursuit of prey (Exploration phase):

In this part of the rats' chasing and fighting behavior, the rats' exploration mechanism is described. Rats have powerful eyes that allow them to track and detect their prey, but sometimes the prey may not be visible.

Due to their social behavior, rats often hunt in groups, which makes them highly effective at locating and capturing their prey. To model this behavior, we assume that the best searcher knows where the prey is located, and the other searchers can update this information based on their observations.

This mechanism is described quantitatively using the following equations:

$$Loc = \delta \times Loc_i + \beta \times (loc_{Best} - loc_i) \quad (3)$$

Where loc_{Best} denotes the best optimal solution and loc_{i+1} the locations of the rats. However, the parameters $\boldsymbol{\delta}$ and $\boldsymbol{\beta}$ are determined as follows:

$$\delta = R - \rho\left(\frac{R}{Max_{Iteration}}\right) \ . \ 1 \le R \le 5 \quad (4)$$

$$\rho = 1.2.3. \ldots . Max_{Iteration} \quad (5)$$

Therefore, throughout the iteration, parameters $\boldsymbol{\delta}$ and $\boldsymbol{\beta}$ are sensitive to good exploration and exploitation, while and are random values between [1. 5] and [0. 2].

Fighting prey (exploitation phase):

The rats attack the target prey detected in the previous phase. However, the prey often tries to escape dangerous situations or defend itself against this attack.

In this case, a deadly battle ensues between the rats and the prey. In some cases, the battle ends with the death of some rats.

Therefore, the fight between the rats and their prey is mathematically described by the formula below:

$$Loc_{i+1} = |Loc_{best} - Loc_i| \quad (6)$$

where Y_{i+1} represents the most recently updated location of the rat. The ideal solution is saved, and the locations of the other search agents are changed relative to it.

In general. The RSO algorithm is presented as follows:

Algorithm 1 Basic Discrete RSO

Output: Optimal solution
Input: the initial rat population P_i Initialize RSO parameters: A, C, and R.
Initialize the rat's populationP_i where i = 1, 2
Now, calculate the fitness value of each search agent.
Choose the best agent fitness value P_{Best}.
while $(k < Max_{Iteration})$ **do**
  for each agent search do
    `Update the positions of current search agents Eq. (3)`
  end for
  Update RSO parameters: A, C, and R.
  Check whether any search agent goes beyond the boundary limit $ofasearchspaceandthenamendit.$
  Calculate the fitness of each search agent $UpdateP_{Best}$ if there is a better solution than the previous optimal solution.
  $k \leftarrow k + 1$.
end while
Return P_{Best}.

4.2. Discrete RSO method to solve the TSP problem

The RSO (Rat Swarm Optimization) method is designed to solve continuous optimization problems. Therefore, it cannot be directly applied to solve discrete combinatorial optimization problems such as the traveling salesperson problem. To use the RSO method for the TSP, several modifications must be made to the algorithm to account for the discrete nature of the problem. These modifications may include changes to the search space, the evaluation function, and the exploration and exploitation strategies used by the algorithm.

In the context of the traveling salesperson problem, each possible route can be represented as a list of cities or as a graph, with each city represented as a vertex and the edges between cities representing the distances between them (as shown in Figure 2).

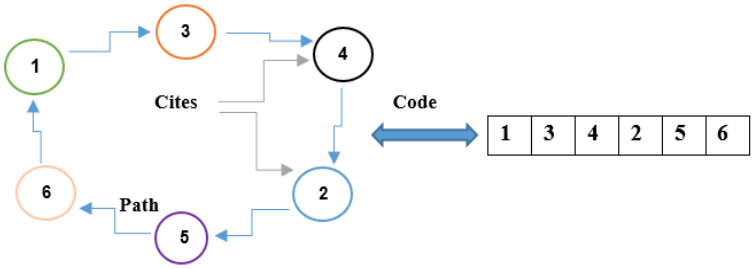
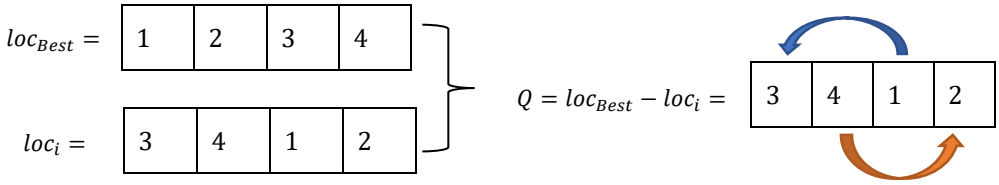


Figure 2. Example of a TSP trip

In the context of using the RSO method to solve the traveling salesman problem, each rat can be associated with a random path (sequence of cities) representing a potential solution. During the optimization process, each rat's movement can involve making small changes or permutations to the order of cities in the path, applying minimal modifications to the current solution. The "fighting with the prey" process can be defined as a verification of the solution, where the solution is accepted if the rat wins and ignored if it does not. This can help to ensure that the algorithm can explore the search space effectively and find good solutions to the TSP.

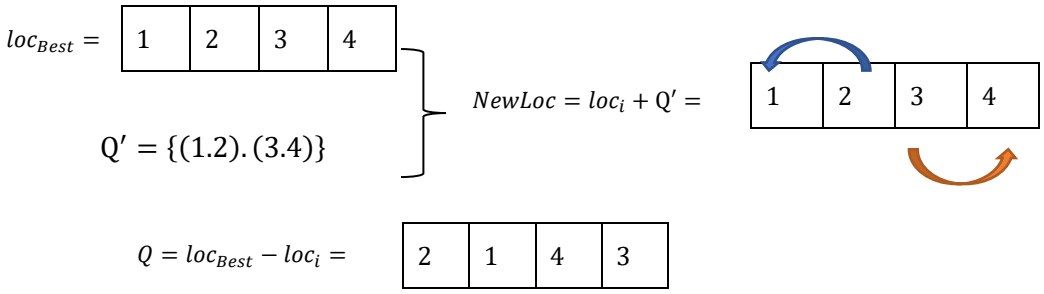
To adapt the RSO method for use with discrete combinatorial optimization problems such as the traveling salesperson problem, the continuous operators used in the original algorithm must be replaced with discrete counterparts. For example, the subtraction operator, which calculates the difference between two positions in the search space, can be defined as a set of permutations that can be performed on one of the positions to obtain a new closer to the other position. This allows the algorithm to explore the space of possible solutions and make changes to the current solution in a way appropriate for the problem's discrete nature.

Example: the subtraction between two positions $loc_{Best} - loc_i$ be defined as the set of permutations to be performed loc_i to obtain a new position loc_{Best}.



$$Q = loc_{Best} - loc_i = \{(3.1).\ (2.4)\}$$

Similarly, the addition operator in the RSO method can be adapted for use with discrete optimization problems by defining it as a set of permutations that can be applied to a path (list of cities to visit) to modify the current position. This operator allows the algorithm to make changes to the current solution in a way that is appropriate for the problem's discrete nature and can help explore the space of possible solutions more effectively. For example, the operator could swap the positions of two cities in the path or insert a new city into the path at a specific position. Again, these changes can help the algorithm explore the search space and find good solutions to the TSP.



Finally, the multiplication operator in the RSO method can be defined as an operator that allows reducing the number of permutations applied to a path. This operator can be applied between a real number and a permutation list, allowing the algorithm to make more targeted changes to the current solution and avoid making unnecessary permutations.

- $\beta \times (loc_{Best} - loc_i)$.
- $Q' = (loc_{Best} - loc_i) = \{(1.2). (3.4)\}$
- $\beta = 0.5$
- $\beta \times (loc_{Best} - loc_i) = \{(1.2)\}$

To improve the quality of solutions to the traveling salesperson problem, various neighborhood search strategies can be used. One example of a reliable neighborhood search method for the TSP is the two-exchange function, which involves making small changes to the order of cities in the current solution to find a better solution.

In this work, we will present two versions of the rat swarm optimization (RSO) algorithm for solving the TSP: the basic DRSO (Discrete Rat Swarm Optimization) algorithm and the improved hybrid HDRSO (Hybrid Discrete Rat Swarm Optimization) algorithm. The basic DRSO algorithm uses the RSO method in its original form, while the HDRSO algorithm incorporates additional techniques and modifications to improve its performance.

Before presenting the developments and modifications to the RSO method, we will introduce the basic version of the algorithm and discuss its limitations.

4.3. The limits of the basic discrete rat swarm optimizer

The Discrete Rat Swarm Optimization (DRSO) algorithm, introduced by Mzili et al. (2022), emulates the behavior of rats in the wild, where they collaborate to search for prey. The population of the algorithm consists of rats, one of which is designated as the captain, responsible for leading the group to the prey. However, we found that the algorithm tends to converge to a local optimum after several iterations, which hurts its performance (as shown in Table 2).

This behavior is because the captain rat does not have accurate information about the position of the prey, which can lead the whole group to be trapped in a false location, as rats do in the wild. Unfortunately, the captain is not replaced periodically, and the whole group suffers from having to start the search from the beginning, resulting in considerable delays.

To address these limitations, we propose an improved hybrid version of the RSO algorithm that incorporates additional strategies and modifications to improve its

performance. One strategy is to search for additional generations of rats during the prey search phase that can fight the prey and are more efficient, with varying information about the location of the prey through additional enhancement, growth and selection heuristics. The rats work together to guide the population to the optimal solution, and if a captain is trapped, another rat can take over.

In addition, we suggest introducing a random mutation operator to allow the population to explore new solutions and escape from local optima. Applying this operator with a low probability avoids interfering with the algorithm's convergence. Moreover, we propose to incorporate a local search mechanism to refine the solutions found by the algorithm. This mechanism can be applied to the best solutions the algorithm finds to improve their quality further. The proposed hybrid version of the RSO algorithm combines multiple strategies to address its limitations, increase population diversity, improve exploration capabilities, and refine the algorithm's solutions.

Table 2. Results of the basic DRSO algorithm

Instance	Opt	Best	Average	Worst	PDav	PDbest	Time(s)	SD
eil51	426	426	432.57	442	1.54	0	6.76	4.34
berlin52	7542	7542	7788.79	8111	3.27	0	3.84	181.9
st70	675	675	685.46	699	1.55	0	9.39	7.68
Olivier30	420	420	421.62	426	0.38	0	0.39	2.72
eil76	538	549	569.5	591	5.85	2.00	13h35	16 775
kroA100	21282	21353	21748.4	21986	2.19	0.33	18.76	204.90
kroB100	22141	22337	23165.5	23929	2.62	0.87	9.50	644.91
Eil 101	629	653	672.62	696	6.93	3.67	0.39	15.17
ch130	6 110	6275	6507	6816	6.49	2.62	13.53	175.30
Rat99	1211	1229	1274.44	1308	5.23	1.46	0.39	27.98
D198	15780	16045	16517.8	16994	4.67	1.65	19.89	327.17

5. Proposed hybrid discrete rat swarm optimizer

To improve the basic version of the RSO algorithm and address its limitations, we propose incorporating a mechanism common in many animal species: mating and selection. To improve the next generation of any animal breed, animals (male or female) will choose their life partner based on specific characteristics that are distinctive for that type of animal to ensure the continuation of the desired qualities. For example, in wild animals such as lions and wolves, females will choose the strongest males, and sometimes this selection is made through mortal combat. On the other hand, in animals that value aesthetics to preserve the beauty of the offspring, females will choose the most beautiful, attractive, and elegant males.

We adopt this mechanism in our rat swarm optimization algorithm, where rats can mate after selecting the most intelligent and strongest elements that can find the position of the prey and successfully attack it without dying. This mechanism maintains the characteristics of the swarm and the group and generates stronger solutions. Additionally, at each iteration, we can incorporate basic local improvement heuristics such as 2-opt and 3-opt (Zhong, 2021).

However, the algorithms derived from the K-Opt algorithm have high complexity in terms of time $\theta(n^k)$ and memory usage, so caution must be taken when using them. For this reason, we have associated these algorithms with probability. A new random parameter T is added at this level, and its value is between [0, 1]. We can choose which operator to call at each iteration based on this value. This allows us to balance the exploration and exploitation of the search space and improve the algorithm's overall performance.

5.1. Crossover and selection operators

Crossover operations were first introduced in genetic algorithms to create a new population. The idea behind crossover is to use each parent's best qualities through a new production generation. Various crossover operators have been proposed in the literature to solve the traveling salesman problem. This paper adopts the modified unified intersection to improve the search strategy of DRSO. In this paper, the crossover is performed as follows:

1) A son is matched with the best-found father, which can also be named *G*best to generate another individual who can be better than both.
2) A and B, representing different solutions to the traveling sales problem, are selected as parents for the crossover operation.
3) The modified unified intersection operator is applied to A and B to generate a new individual, C. This operator combines the strengths of both parents and generates a new solution that is better than both original solutions.
4) C is then compared to the current *G*best, which represents the best solution found so far by the algorithm, and if it is better, it becomes the new *G*best.
5) This process is repeated until a new population of solutions is generated, with each new solution being generated through the RSO Crossover operator.

The advantage of using the RSO Crossover operator is that it allows for a more efficient and effective search strategy for solving the traveling sales problem. By combining the strengths of both parents and generating a new individual with the best qualities of both, the algorithm can explore a wider range of solutions and find better solutions more quickly. This leads to improved performance and better solutions to the traveling sales problem.

5.2. 2-Opt move algorithm

The 2-opt algorithm is a local search algorithm that is commonly used to improve the quality of solutions to the traveling sales problem. It works by iteratively removing and reconnecting pairs of edges in the current solution to improve the cost of the solution. This is done by removing intersecting edges and applying the triangle inequality to ensure the solution is valid.

The 2-opt algorithm is applied in the last step of the RSO Crossover operator after a new population of solutions has been generated. This allows the algorithm to further improve the quality of the solutions by removing any remaining intersections and ensuring that the solutions are valid. Overall, using the RSO Crossover operator combined with the 2-opt algorithm allows for a more efficient and effective search strategy for solving the traveling sales problem.

In Figure 3, the route <a; b; c; d> is changed to <a; c; b; d> by reversing the order of visiting cities b and d. This involves removing the edges (a, b) and (c, d) and reconnecting them in the new order (a, c) and (b, d).

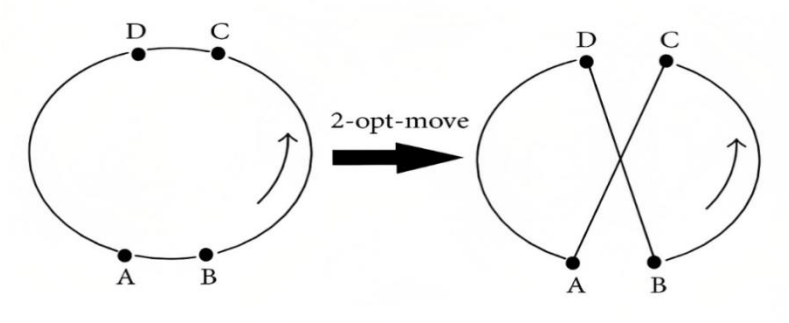


Figure 3. 2-Opt move

The 2-opt algorithm is a special case of the more general k-opt algorithm, where k is the number of edges that are removed and reconnected at each step. The 2-opt algorithm specifically considers k = 2, meaning that only two edges are swapped at each step (Figure 4). While it is possible to generalize the 2-opt algorithm to consider higher values of k, such as the 3-opt (Figure 5) algorithm, which considers k = 3, this is generally not necessary. This is because increasing k leads to a more complex search space and a higher computational cost without necessarily significantly improving the quality of the solutions.

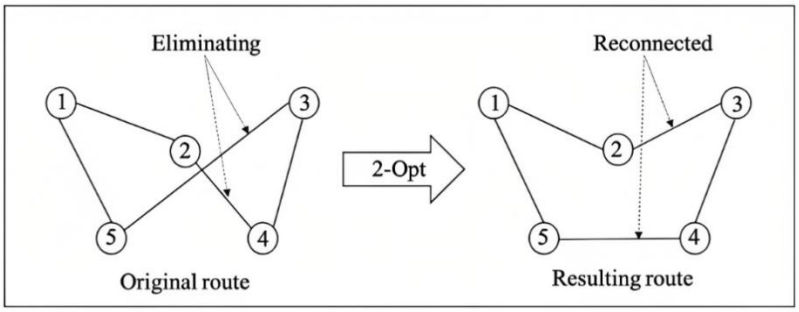


Figure 4. 2-Opt swap example

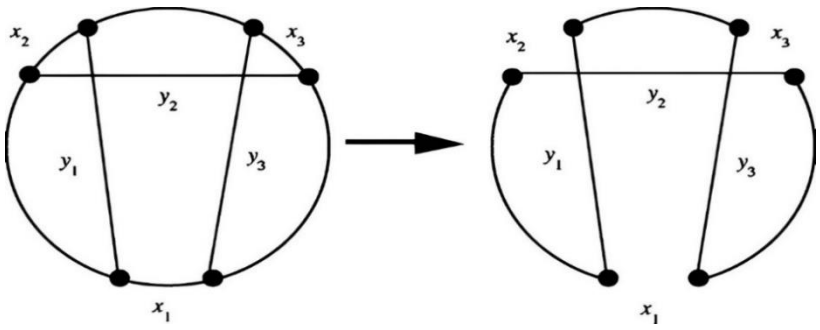


Fig.5. 3-Opt swap example

5.3. Modification and adjustment of basic parameters

Our approach has several variables that play important roles in the optimization process.

- C is a variable that helps to correctly explore the discrete case's search space correctly. It determines the number of permutations performed on a path (city sequence) to find a potentially optimal solution. If C < 1, we can generate a different path. If C=1, we get the same path.
- δ is a variable that controls the adjustment of the objective function in the continuous case or the modification of the entire trajectory in the discrete case.
- T is a variable that plays a crucial role in balancing the use of auxiliary operators. At each iteration, T takes on a random value.
  - T < 0.5, we perform permutations on the path according to a certain equation.
  - 0.5 <= T <= 0.9, we select a solution for the crossover using the crossover operator (a crossover between a solution and the best solution found so far).

T > 0.9, we apply the local heuristic 3-OPT, which has high complexity in terms of time and memory but can quickly converge to a local optimum. We call it a low probability due to its high complexity.

Algorithm 2 : Hybrid HDRSO Algorithm

```
Output: Optimal solution
Input: the initial rat population P_i
Initialize RSO parameters: A, C, and R.
Initialize the rat's population P_i where i = 1, 2
Now, calculate the fitness value of each search agent.
Choose the best agent fitness value P_Best.
while (k < Max_Iteration OR Best result obtained) do
    For each search agent do
    2 − Opt(nb_city, P_i)
    T ← Rand();                          ▷ Initialize the new parameter
    if T < 0.5 then
        Update the positions of current search agents Eq. (3)
    else if T ≤ 0.9 then
        cross-over (P_i, P_Best)
    else
        3 − Opt(nb_city, P_i);
    end if
    Update RSO parameters: A, C, and R.
    Check whether any search agent goes beyond the boundary limit
       of a search space and then amend it.
    Calculate the fitness of each search agent UpdateP_Best if there is
        a better solution than the previous optimal solution.
    k ← k + 1.
end while
Return P_Best.
```

6. Experimental results and comparison

The basic and improved hybrid algorithms, HDRSO (Hybrid Discrete Swarm Optimizer Search), are applied to solve the traveling salesperson problem. They are tested on several TSP instances (benchmarks) from the public electronic library TSPLIB of TSP problems. Each TSP instance provides a list of cities with their coordinates (x and y) (as shown in Figure 6).

```
NAME : Berlin52
TYPE : TSP
DIMENSION : 52
EDGE_WEIGHT_TYPE : EUC_2D
NODE_COORD_SECTION
1 565.0 575.0
2 25.0 185.0
3 345.0 750.0
4 945.0 685.0
.
.
.
52 1740.0 245.0
```

Figure 6. Example of berlin52 instance

The instance name represents its name in TSPLIB concatenated with the number of cities in the instance: the instance named st70 has 70 cities. Euclidean distances of these TSP instances are used for experiments and comparisons. The basic and improved algorithm is tested using C++ as a programming language under the 64-bit Windows 10 operating system. The tests are performed on a Dell laptop with a 2.00 GHz Intel Core i5 processor and 16 GB of RAM. The values of the parameters of the proposed algorithm are chosen based on some preliminary tests. Then, the comparison is made based on the following criteria: 1) The Best Value, which designates the best solution obtained by each algorithm; 2) The mean value designates the average value of the 20 solutions obtained after 20 executions of an algorithm; 3) The worst value is the worst value obtained by an algorithm; 4) PDav(%) designates the percentage of deviation of the average length of the solution from the optimal solution of 20 executions:

$$\frac{\text{average-opt}}{\text{opt}} \times 100 \tag{7}$$

5) The STD is the standard deviation; finally, 6) the Time value is the average time in seconds of the 20 executions.

The proposed HDRSO algorithm is compared with the basic algorithm and five recently developed bio-inspired metaheuristics: DJAYA, RNN-SA, GGSC-SSA, DSSA, and DSOS. The DJAYA algorithm (Gunduz & Aslan, 2021) is a population-based approach proposed to solve constrained and unconstrained optimization problems. RNN-SA (Rahman & Parvez, 2021) is an extension of the well-known Nearest Neighbor algorithm, designed to build routes efficiently. GGSC-SSA (Wu et al., 2021) is inspired by the foraging behavior of sparrows, while DSSA (Bas & Ülker, 2021) is based on the behavior of spiders. Finally, DSOS (Ezugwu & Adewumi, 2017) is a metaheuristic algorithm that takes inspiration from the symbiotic interactions among organisms in nature.

The performance of these algorithms is compared on a set of TSP instances from the TSPLIB library using the Euclidean distance. By comparing the algorithms on this benchmark dataset, we can gain insights into their relative performance and efficiency for solving the traveling salesperson problem.

Table 3. Initial parameter for HDRSO

parameter	value
Population size	100
R	Radom N in [1. 5]
C	Radom N in [0. 1]
T	Radom N in [0. 1]
$Max_{Iteration}$	1000

6.1. Comparison between the base DRSO result and the HDRSO

In this section, we will compare the basic version of DRSO with the HDRSO and examine the impact of altering parameters and introducing a new type of motion.

Table 4 compares the results of the basic DRSO and the HDRSO.

Table 4. Comparison between the hybrid version of DRSO and the base version

						Basic DRSO		
Instance	Opt	best	Average	PDav	PDbest	SD	Time(s)	Best
eil51	426	426	432.57	1.54	0	4.34	6.76	426
berlin52	7542	7542	7788.79	3.27	0	181.9	3.84	7542
st70	675	675	685.46	1.55	0	7.68	9.39	675
Oliver30	420	420	421.62	0.38	0	2.72	0.39	420
seil76	538	549	569.5	5.85	2.00	16.775	13.35	538
kroA100	21282	21353	21748.4	2.19	0.33	204.90	18.76	21282
kroB100	22141	22337	23165.5	2.62	0.87	644.91	9.50	22141
eil101	629	653	672.62	6.93	3.67	15.17	0.39	633
ch130	6.110	6275	6507	6.49	2.62	175.30	13.53	6105
Rat99	1211	1229	1274.44	5.23	1.46	27.98	0.39	1211
D198	15780	16045	16517.8	4.67	1.65	327.17	19.89	15874

Table 4 (Continuous). Comparison between the hybrid version of DRSO and the base version

						HDRSO				
Instance	Opt	best	Average	PDav	PDbest	SD	Time(s)	T-value	Sig	WSR
eil51	426	426	426.5	0.11	0	0.53	5.63	16.67	+++	=
berlin52	7542	7542	7542	0	0	0	0.8	6.07	+++	=
st70	675	675	676.25	0.18	0	2.37	8.72	14.55	+++	=
Oliver30	420	420	420	0	0	0	0.11	2.67	++	=
eil76	538	549	541.33	0.61	0	4.17	10.57	7.29	+++	+
kroA100	21282	21353	21295	0.06	0	28.77	28.55	28.77	+++	+
kroB100	22141	22337	22142.78	0.01	0	5.33	14.75	7.09	+++	+
eil101	629	653	637.6	1.36	0.63	4.44	34.5	9.91	+++	+
ch130	6.110	6275	6143	0.54	-0.08	42.73	44.89	9.02	+++	+
Rat99	1211	1229	1212.25	0.10	0	1.25	34.04	9.93	+++	+
D198	15780	16045	15956	1.11	0.95	69.93	101.3	7.51	+++	+

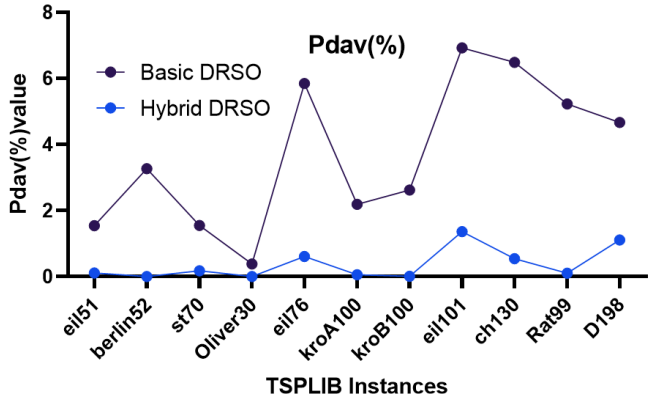


Figure 7. Convergence PDAV(%) graph of basic and hybrid DRSO

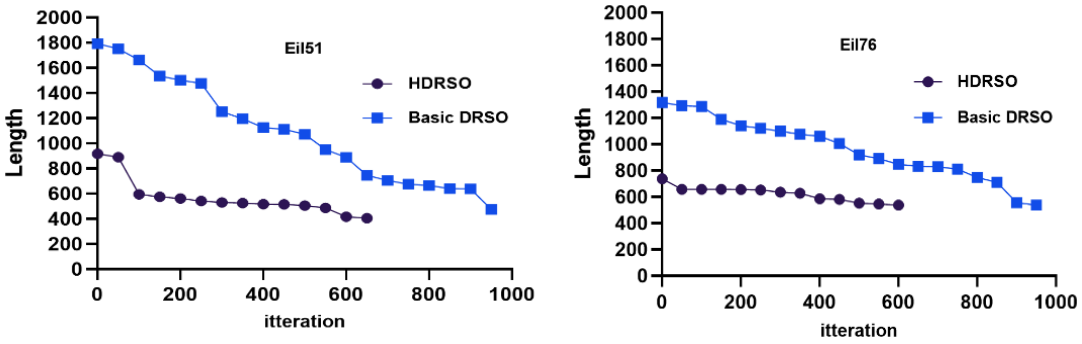


Figure 8. Comparison of the number of iterations necessary to obtain the best solution for eil51(opt=426) and Eil76(opt=538)

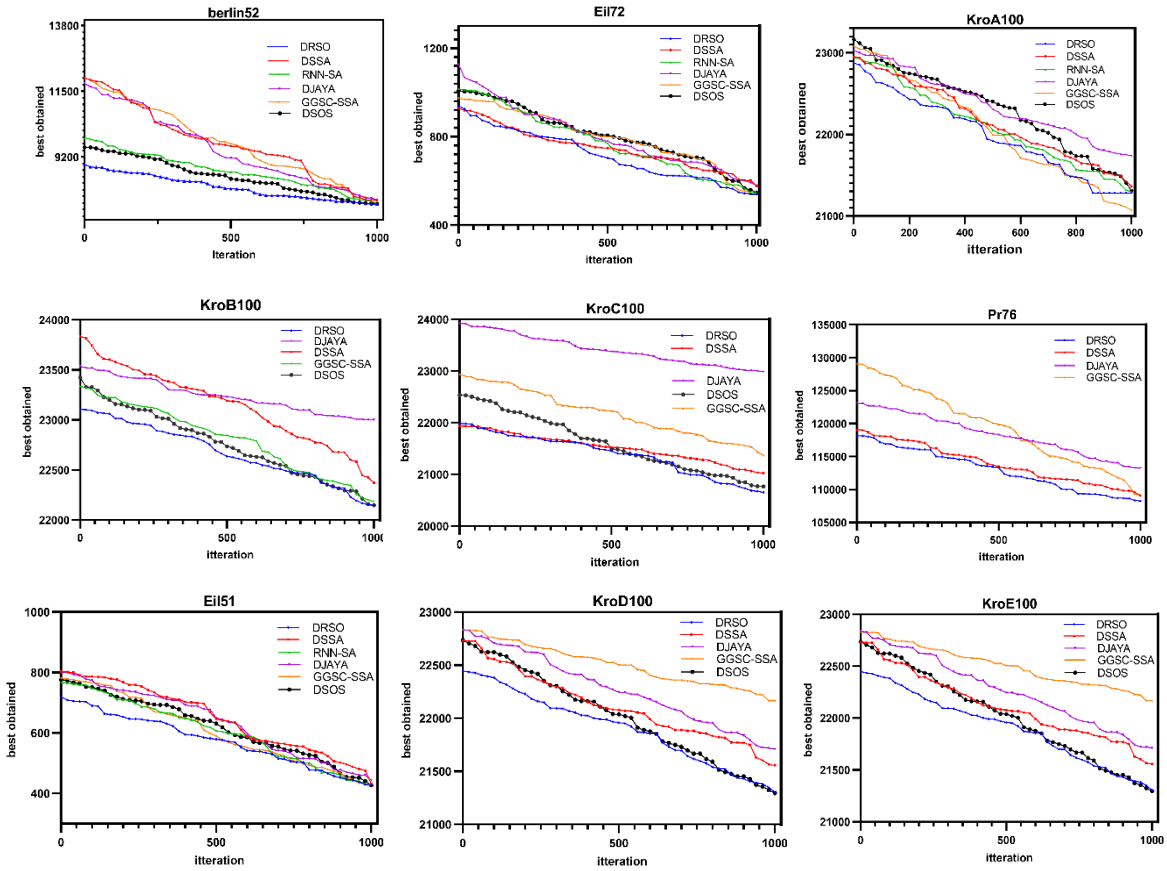


Figure 9. Average convergence curves of all comparison algorithms

6.2. Comparison between the HDRSO and other recently developed metaheuristics

This section will compare the hybrid HDRSO metaheuristic with several other techniques developed in 2020 to evaluate its ability to solve TSPLIB instances. To do this, we ran HDRSO on 26 TSPLIB instances with 20 independent runs and used a parametric test to analyze the results. It's worth noting that the experiments were conducted on different computers and platforms, so we will not be making runtime comparisons.

In Experiment 1, HDRSO is compared with the basic DRSO and DJAYA, RNN-SA, GGSC-SSA, DSSA, and DSOS on a set of 26 symmetric TSP instances. The results of this comparison are listed in Tables 4-9, which show the mean, best value, standard deviation, and average performance time for each algorithm. The best results and averages are highlighted in bold.

To determine significant differences between the results, we performed a student's t-test for each algorithm compared to HDRSO. The t-values were computed using the standard deviation and mean of 20 independent runs for each problem.

The t-test results can be found in the "T-value" and "Sig." columns of the tables.

To test for significant differences between HDRSO and the other techniques, we derived five significant levels using critical values at the 95% confidence level (t0.05 =

1.960) and the 99% confidence level (t0.01 = 2.576). The "Sig." significance levels are defined as follows:

- t > 2.756: +++ (Extremely significant)
- 1.960 < t ⩽ 2.756: ++ (Significant)
- 0 < t ⩽ 1.960: + (Slightly significant)
- t = 0: = (Equal)
- 1.960 ⩽ t < 0: - (Insignificant)

The WSR column represents the sign of the difference between each pair of paired observations. The smaller of the two rank sums, one for positive values WSR+ and one for negative values WSR-, is used as the test statistic for Wilcoxon hypothesis tests.

Table 5. Comparison of the experimental results of the HDSRO with the DSSA

		HDRSO					DSSA (2021)						
Instance	Opt	Best	Average	PDav(%)	SD	Time	Best	Average	PDav(%)	SD	T-value	sig	WSR
Eil51	426	426	426.5	0.11	0.53	5.63	431.87	431.87	0.65	2.8	8.43	+++	+
Berlin52	7542	7542	7542	0	0	0.8	7659	7659	1.55	117	4.47	+++	+
Eil76	538	538	541.33	0.61	4.17	10.57	559.31	559.31	3.96	0.4	19.19	+++	+
Pr76	108 159	108159	108206.2	0.04	82.51	8.76	108880	108 880	0.66	721	4.15	+++	+
Rat99	1211	1211	1212.25	0.10	1.25	34.04	1221	1221	0.82	dix	3.88	+++	+
Rd100	7910	7905	7916.6	0.08	17h55	19.69	8120	8120	2.65	210	4.35	+++	+
KroA100	21 282	21282	21295	0.06	28.77	28.55	21363	21 363	0.38	81	3.54	+++	+
KroB100	22 141	22141	22142.78	0.008	5.33	14.75	22347	22 347	0.93	206	4.43	+++	+
KroC100	20 749	20749	20752.33	0.01	8.16	14.52	20997	20 997	1.19	248	4.41	+++	+
KroD100	21 294	21294	21313.89	0.09	38.58	21.82	21552	21 552	1.21	258	4.08	+++	+
KroE100	22 068	22068	22113	0.20	30.27	31.92	22407	22 407	1.53	339	3.86	+++	+
Lin105	14 379	14379	14379	0	0	15h44	14502	14 502	0.85	123	4.47	+++	+
Pr107	44 303	44303	44351.67	0.10	80.56	30.08	44346	44 346	0.09	43	0.28	+	+
Pr124	59 030	59030	59039.2	0.01	19.39	13.07	59087	59 087	0.09	57	3.56	+++	+
Pr136	96 772	96785	97128.3	0.36	414	65.21	103460	103 460	6.91	6688	4.23	+++	+
Pr144	58 537	58537	58537	0	0	24.73	58669	58 669	0.22	132	4.47	+++	+
KroA150	26.524	26579	26741	0.81	144.10	40.24	27.027	27.027	1.59	503	2.44	++	+
Pr152	73.682	73682	73761.45	0.10	222.19	14.33	74.462	74.462	1.05	780	3.86	+++	+
Rat195	2323	2344	2360.71	1.62	12.53	114.81	2353	2353	1.29	30	1.06	+	+

Table 6. Comparison of experimental results of HDSRO with RNN-SA

		HDRSO					RNN-SA (2021)						
instances	opt	Best	Average	PDav(%)	SD	Time	Best	Average	SD	PDav(%)	T-value	Sig	WSR
eil51	426	426	426.5	0.11	0.53	5.63	428.87	429.44	0.52	0.81	17.7080	+++	+
berlin52	7542	7542	7542	0	0	0.8	7544.37	7544.37	1.3	0.03	8.1530	+++	+
st70	675	675	676.25	0.18	2.37	8.72	677.11	679.33	3.04	0.64	3.5734	+++	+
eil76	538	538	541.33	0.61	4.17	10.57	544.37	549.16	4.40	2.07	5.7764	+++	+
rat99	1211	1211	1212.25	0.10	1.25	34.04	1219.24	1229.29	5.68	1.51	13.1029	+++	+
kroA100	21 282	21282	21295	0.06	28.77	28.55	21285.44	285.44	0.20	0.02	1.4860	+	+
kroE100	22 068	22068	22113	0.20	30.27	31.92	22119.90	164.92	58.65	0.44	3.5180	+++	+
eil101	629	633	637.6	1.36	4.44	34.5	644.92	652.31	5.39	3.71	9.4204	+++	+
bier127	118 282	118599	118681.3	0.33	90.45	42.16	119331.15	849.69	294.00	1.33	16.9870	+++	+
ch130	6110	6105	6143	0.54	42.73	44.89	6171.87	6249.18	66.31	2.28	6.0195	+++	+

		HDRSO					RNN-SA (2021)						
instances	opt	Best	Average	PDav(%)	SD	Time	Best	Average	SD	PDav(%)	T-value	Sig	WSR
pr136	96 772	96785	97128.3	0.36	414	65.21	96922.41	100 335.16	1979.60	3.68	7.0912	+++	+
ch150	6528	6548	6560.16	0.49	12.92	49.66	6552.30	6553.94	1.58	0.40	2.1371	++	+
kroA150	26 524	26579	26741	0.81	144.10	40.24	26821.83	27008.24	194.18	1.83	4.9425	+++	+
kroB150	26.130	26137	26183.5	0.20	80.74	51.83	26327.09	26.577.15	232.5	1.71	7.1528	+++	+
pr152	73.682	73682	73761.45	0.10	222.19	14.33	73843.09	74669.58	836.59	1.34	4.6919	+++	+
u159	42 080	42080	42140.75	0.14	119.06	12.35	42162.75	547.66	90.57	1.11	12.1647	+++	+
rat195	2323	2344	2360.71	1.62	12.53	114.81	2348.70	2361.50	11h49	1.66	0.2078	+	+
kroA200	29 368	29434	29525.67	0.53	86.55	82.03	29541.83	29626.42	73.01	0.88	3.9792	+++	+
kroB200	29 437	29479	29509.33	0.24	34.11	121.4	29825.16	30033.03	163.02	2.02	14.0622	+++	+
pr264	49 135	49135	49135	0	0	69.19	49197.32	49375.75	232.85	0.49	4.6239	+++	+
lin318	42 029	42253	42706.67	1.46	343.37	130.33	42862.50	43041.10	122.01	2.71	4.1043	+++	+

Table 7. Comparison of experimental results of HDSRO with DJAYA

		HDRSO						DJAYA					
instances	opt	Best	Average	PDav(%)	PDbest(%)	SD	Time	Best	PDbest(%)	SD	T-value	Sig	WSR
oliver30	420	420	420	0	0	0	0.39	426.88	0.74	2.74	11.2293	+++	+
eil51	426	426	426.5	0.11	0	0.53	5.63	440.18	2.64	4.95	12.2891	+++	+
berlin52	7542	7542	7542	0	0	0	0.8	7580	0.48	80.60	2.1085	++	+
st70	675	675	676.25	0.18	0	2.37	8.72	702.30	3.72	9.56	11.7177	+++	+
eil76	538	538	541.33	0.61	0	4.17	10.57	573.17	5.10	6.33	18.7851	+++	+
pr76	108 159	108159	108206.2	0.04	0	82.51	8.76	113258.29	4.71	1711.93	13.1825	+++	+
eil101	629	633	637.6	1.36	0.63	4.44	34.5	677.37	5.46	4.87	26.9882	+++	+
ch150	6 528	6548	6560.16	0.49	0.30	12.92	49.66	6638.63	1.63	52.79	6.4571	+++	+
kroa100	21 282	21282	21295	0.06	0	28.77	28.55	21735.31	2.13	331.33	5.9208	+++	+
krob100	22 141	22141	22142.78	0.008	0	5.33	14.75	22973.73	3.76	234.79	15.8233	+++	+
kroc100	20 749	20749	20752.33	0.01	0	8.16	14.52	21702.02	4.59	186.32	22.7731	+++	+
krod100	21 294	21294	21313.89	0.09	0	38.58	21.82	22631.25	6.28	487.62	12.0443	+++	+
kroe100	22 068	22068	22113	0.20	0	30.27	31.92	22582.47	2.33	252.07	8.2698	+++	+

Table 8. Comparison of experimental results of HDSRO with GGSC-SSA

		HDRSO						GGSC-SSA(2021)			
Instance	Opt	Best	Average	PDav(%)	PDbest(%)	SD	Time(s)	Best	PDav(%)	Time(s)	WSR
ulysses22	75.67	71	71.8	-5.09	-6.57	0.40	0.003	75.24	0.00	1.95	+
eil51	426	426	426.5	0.11	0	0.53	5.63	426	0.00	3.87	=
berlin52	7542	7542	7542	0	0	0	0.8	7542	0.00	9.48	=
st70	675	675	676.25	0.18	0	2.37	8.72	676.96	0.29	8.26	+
pr76	108159	108159	108206.2	0.04	0	82.51	8.76	109170.3	0.94	15.13	+
eil76	538	538	541.33	0.61	0	4.17	10.57	539.79	0.33	9.32	+
rat99	1211	1211	1212.25	0.10	0	1.25	34.04	1211	0.00	11.4	=
kroA100	21282	21282	21295	0.06	0	28.77	28.55	20989.04	0.00	19.88	+
kroB100	22140	22141	22142.78	0.008	0	5.33	14.75	22152	0.05	11.5	+
kroC100	20749	20749	20752.33	0.01	0	8.16	14.52	21346.53	2.88	11.92	+
kroD100	21294	21294	21313.89	0.09	0	38.58	21.82	22138.53	3.97	16.48	+
rd100	7910	7905	7916.6	0.08	-0.06	17.55	19.69	7951.28	0.52	15.34	+
eil101	629	633	637.6	1.36	0.63	4.44	34.5	638	1.43	11.75	+
lin105	14379	14379	14379	0	0	0	15.44	14543.56	1.14	15.95	+

pr124	59030	59030	59039.2	0.01	0	19.39	13.07	59607.74	0.98	17.68	+
ch130	6110.86	6105	6143	0.54	-0.08	42.73	44.89	6115.25	0.07	15.59	+
pr144	58537	58537	58537	0	0	0	24.73	58862.09	0.56	20.05	+
d198	15780	15874	15956	1.11	0.95	69.93	101.3	15951.29	1.09	74.85	+
kroA200	29368	29434	29525.67	0.53	0.22	86.55	82.03	29507.35	0.47	49.31	+
kroB200	29437	29479	29.509.33	0.24	0.14	34.11	121.4	29678.92	0.82	64.37	+
pr226	80369	80369	80372.73	0.004	0	12.36	27.04	81361.17	1.23	35.63	+
lin318	42090	42253	42706.67	1.46	0.38	343.37	130.33	43023.73	2.22	184.73	+
fl417	11861	11857	11876	0.12	-0.03	20.84	212.95	11936.25	0.63	257.88	+
pr439	107217	108647	109 687.5	2.30	1.31	1124.87	226.31	113074.47	5.46	86.09	+

Table 9. Comparison of the experimental results of the Hybrid DSRO with the DSOS.

		HDRSO						DSOS (2017)				
Instance	Opt	Best	Average	PDav(%)	PDbest(%)	SD	Time(s)	Best	Average	PDav(%)	PDbest(%)	WSR
eil51	426	426	426.5	0.11	0	0.53	5.63	426	427.90	0.45	0	=
berlin52	7542	7542	7542	0	0	0	0.8	7 542	7 542.60	0.01	0	=
st70	675	675	676.25	0.18	0	2.37	8.72	675	679.20	0.62	0	=
eil76	538	538	541.33	0.61	0	4.17	10.57	542	547.40	1.75	0.74	+
rat99	1211	1211	1212.25	0.10	0	1.25	34.04	1235	1 240.74	2.46	1.98	+
kroA100	21 282	21282	21295	0.06	0	28.77	28.55	21282	21 409.50	0.60	0	=
kroB100	22 140	22141	22142.78	0.008	0	5.33	14.75	22140	22 339.20	0.90	0	-
kroC100	20 749	20749	20752.33	0.01	0	8.16	14.52	20749	20 881.60	0.64	0	=
kroD100	21 294	21294	21313.89	0.09	0	38.58	21.82	21294	21 493.10	0.94	0	=
kroE100	22 068	22068	22113	0.20	0	30.27	31.92	22068	22 231.10	0.74	0	=
eil101	629	633	637.6	1.36	0.63	4.44	34.5	640	650.60	3.43	1.75	+
pr107	44 303	44303	44351.67	0.10	0	80.56	30.08	44314	44 445.10	0.32	0.02	+
pr124	59 030	59030	59039.2	0.01	0	19.39	13.07	59030	59 429.10	0.68	0	=
pr136	96 772	96785	97128.3	0.36	0.01	414	65.21	97437	97 673.20	0.93	0.69	+
pr144	58 537	58537	58537	0	0	0	24.73	58565	58 817.10	0.49	0.05	+

6.3. Analysis and discussion

To further verify the effectiveness of the proposed algorithm, we conducted a comparative analysis with several state-of-the-art metaheuristic algorithms published from 2020 to 2022 using a non-parametric statistical test. This allowed us to evaluate the performance of our proposed algorithm and its improvements to these other approaches. The first comparison is with the basic RSO algorithm to see the effects of changes and improvements, followed by a complete comparison with other metaheuristics.

Tables 4-9 show the experimental results of the HDRSO algorithm compared to the DJAYA (2021), DSSA (2021), DSOS (2017), RNNA-SA (2021), and GGSC-SSA (2021) methods. The statistical criteria chosen for this comparison are to show the difference in the obtained values, the mean, the deviation, and the computation time. The tables show that our HDRSO algorithm obtained the optimum several times compared to the other metaheuristics. The minimal values of STD and Pdav(%) can justify that HDRSO could obtain the optimum several times compared to the other metaheuristics. In the first table, we can see that the statistical values are minimal in the improved version.

This indicates that the hybridization and improvement strategy chosen was able to create a very robust new algorithm that could solve many other combinatorial problems.

In this section, we chose to make a statistical evaluation using a student parametric test (T-test) by comparing the results of HDRSO and the Basic DRSO. The t-test results presented in Table 4 show that HDRSO is 100% superior to the basic DRSO in all test cases (11 out of 11 assessments). In addition, the t-test results between HDRSO and DSSA are also presented individually in Table 5. HDRSO is highly significant in 84.21% (16 out of 19 assessments) and either significantly better in 5.26% (1 out of 19 assessments) or slightly better in 10.52% (2 out of 19 assessments) of the cases, as reflected in the differences in results. Table 6 shows that HDRSO outperforms RNN-SA in 86.22% (19 out of 22 evaluations) of the cases and is either significantly better in 4.54% (1 out of 22 evaluations) or slightly better in 9.09% (2 out of 22) of the evaluations.

For the comparison with DJAYA, we will see that our algorithm is significantly better than Djaya's in almost all test cases (92.30% or 12 out of 13 evaluations) and 7.69% significantly better (1 out of 13 assessments). The algorithms that do not have enough information to make a statistical comparison according to the student's T-test are compared based on the average time and their ability to obtain the optimal value. Regarding the comparison between HDRSO and GGSC-SSA, we can see that HDRSO obtained the optimum at 70.83% (17 tests out of 24) and exceeded the optimum at 16.66% (4 tests out of 24). On the other hand, GGSC-SSA obtained the optimum at only 16.66% (4 tests out of 24), which is significantly weaker compared to HDRSO.

Finally, in the comparison between HDRSO and DSOS, we can see that HDRSO obtained the optimum in 86.66% (13 tests out of 15), while DSOS obtained the optimum in 60% (9 tests out of 15) with a 26.66% difference compared to HDRSO. Furthermore, when we analyze the average values of PDav(%), we can see that the PDav of HDRSO is lower than that of DSOS at 100% (15 tests out of 15), which can justify that for each test, the solutions of the 20 executions made by HDRSO are very close to the optimum, whereas those of DSOS are not.

In some tests, the HDRSO exceeded the optimal value proposed by the TSPLIB library. Moreover, for four instances (ch130, rd100, fl417, ulysse22), see Table 9. These new values obtained can be new references for future research.

We will also confirm our analysis and comparison with a non-parametric Wilcoxon test (Fix & Hodges Jr, 1955) with a 95% confidence interval (α=0.05) to compare our optimizer and other metaheuristics.

This test was applied to compare the difference between the best-Obt value in two algorithms for comparison and ranking.

N denotes the number of test cases, and W+ represents the scores of the cases with the best performance in the proposed algorithm (sums of WSR+). While W- represents the sum of the scores of the cases where the proposed algorithm performs worse than the comparative algorithm (sums of WSR-), and the p-value is compared to a critical value α =0.05 in the Wilcoxon signed-rank test. If the p-value $\leq \alpha$, it indicates a significant difference between the performance of the two algorithms. However, if the p-value $> \alpha$, then there is no significant difference between the performance of the two algorithms.

Table 10. Results of the Wilcoxon signed-rank test for the comparison of the best-obtained solution of HDRSO with other metaheuristics

Comparison	Dimension N	W-	W+	P-value	Significantly (P < 0.05)?
HDRSO vs Basic	11	-56,00	0	0,016	YES
HDRSO vs DSSA	19	37,00	153	0,018	YES
DRSO vs RNN-SA	21	0	131	<0,001	YES
DRSO vs GGSC-SSA	24	-16	278	<0,001	YES
DRSO vs DSOS	15	-9	75	0,031	YES

In this table 10, the Wilcoxon test allows us to see that the difference between DRSO and the other metaheuristics is statistically significant. According to these evaluations, the proposed algorithm, which uses the hybridization mechanism, crossover operators, and the 2-opt and 3-opt local search algorithms, outperformed other metaheuristics regarding solution quality and the ability to obtain the optimum. The hybrid HDRSO algorithm is a promising approach for solving the TSP and other combinatorial optimization problems.

7. Conclusion

This paper proposes a new optimization algorithm called the hybridized and discrete rat swarm optimization (HDRSO) algorithm. This algorithm is an improved version of the standard rat swarm optimization (RSO) algorithm and has been adapted to solve the Symmetric Traveler Problem (TSP), a combinatorial optimization problem. Our HDRSO algorithm uses new motion types, mathematical operators, and heuristics, such as basic genetics and K-OPT, to reconstruct its population and introduce a new, more intelligent class of RSO. In addition, the algorithm is inspired by natural rat behavior, such as hunting and chasing prey, and has been discretized for improved performance.

We compare the performance of our HDRSO algorithm to several recently developed metaheuristics, including DJAYA, DSSA, DSOS, RNNA-SA, and GGSC-SSA. The comparison results show that our HDRSO algorithm is more efficient than the other methods in solving TSP problems. The main contributions of this work are the development of a new optimization strategy based on group behavior and other robust mechanisms, as well as the use of a local search heuristic to improve the quality of solutions. This new optimization strategy is applied to the traveling salesman problem, and experimental results show that it outperforms classical heuristics in terms of computational efficiency and solution quality. This method can be useful for real-time decision-making in high-volume logistics transportation, especially in complex and dynamic environments. It can help significantly reduce salesmen's working time and travel costs.

The Discrete Rat Swarm Optimization (DRSO) algorithm is effective in solving the Traveling Salesman Problem (TSP). It can be extended to solve a wide range of other combinatorial optimization problems, such as the Quadratic Assignment Problem (QAP), the Vehicle Routing Problem (VRP), the Job Scheduling Problem (JSSP), and the Knapsack Problem (KP).

DRSO offers several advantages that make it well-suited for these problems. First, it excels at handling discrete optimization problems with a large search space where other optimization methods may have difficulty finding optimal solutions. Second, it uses a natural mechanism that mimics rats' behavior in nature, allowing it to avoid

local optima and identify promising solutions. Finally, DRSO can be easily adapted to different problems by adjusting its parameters, such as population size, crossover rate, and mutation rate. Therefore, it is a versatile algorithm that can be applied to various fields, such as logistics, transportation, manufacturing systems, artificial intelligence, and machine learning applications.

In future work, the proposed algorithm can be extended to solve more advanced discrete optimization problems, such as the Quadratic Assignment Problem (QAP), the Job Shop Scheduling Problem (JSSP), and the Vehicle Routing Problem (VRP). In addition, the algorithm can be generalized to handle a larger number of discrete optimization problems. Further studies will evaluate the algorithm's performance on these more complex problems and explore its potential applications in various fields.

Author contributions: Research problem, M.R., T.M. and I.M.; Conceptualization, M.R., T.M. and I.M.; Methodology, M.R., T.M., and I.M.; Formal analysis, M.R., T.M.; Resources, M.R., I.M.; Original drafting, M.R. and T.M.; Reviewing and editing, M.R., T.M.,I.M; Project administration, M.R., T.M.,IM; Supervision, M.R., I.M.

Funding: This research received no external funding.

Data Availability Statement: The data used to support the findings of this study are included within the article.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abualigah, L., Elaziz, M. A., Sumari, P., Khasawneh, A. M., Alshinwan, M., Mirjalili, S., Shehab, M., Abuaddous, H. Y., & Gandomi, A. H. (2022). Black hole algorithm: A comprehensive survey. Applied Intelligence, 52(10), 11892–11915.

Anuar, S., Selamat, A., & Sallehuddin, R. (2016). A modified scout bee for artificial bee colony algorithm and its performance on optimization problems. Journal of King Saud University - Computer and Information Sciences, 28(4), 395–406.

Atashpaz-Gargari, E., & Lucas, C. (2007). Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. 2007 IEEE Congress on Evolutionary Computation, 4661–4667. https://doi.org/10.1109/CEC.2007.4425083

Barbarosoglu, G., & Ozgur, D. (1999). A tabu search algorithm for the vehicle routing problem. Computers & Operations Research, 26(3), 255–270.

BAŞ, E., & ÜLKER, E. (2021). Dıscrete socıal spıder algorıthm for the travelıng salesman problem. Artificial Intelligence Review, 54(2), 1063–1085.

Chung, S. W., & Freund, J. B. (2022). An optimization method for chaotic turbulent flow. Journal of Computational Physics, 457, 111077. https://doi.org/10.1016/j.jcp.2022.111077

Cinar, A. C., Korkmaz, S., & Kiran, M. S. (2020). A discrete tree-seed algorithm for solving symmetric traveling salesman problem. Engineering Science and Technology, an International Journal, 23(4), 879–890.

Cotta, C., Mathieson, L., & Moscato, P. (2018). Memetic Algorithms. In Handbook of Heuristics (pp. 607–638). Springer International Publishing. https://doi.org/10.1007/978-3-319-07124-4_29

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1(1), 53–66.

Ezugwu, A. E.-S., & Adewumi, A. O. (2017). Discrete symbiotic organisms search algorithm for travelling salesman problem. Expert Systems with Applications, 87, 70–78.

Ezugwu, A. E., Shukla, A. K., Nath, R., Akinyelu, A. A., Agushaka, J. O., Chiroma, H., & Muhuri, P. K. (2021). Metaheuristics: a comprehensive overview and classification along with bibliometric analysis. Artificial Intelligence Review, 54(6), 4237–4316.

Ginidi, A., Ghoneim, S. M., Elsayed, A., El-Sehiemy, R., Shaheen, A., & El-Fergany, A. (2021). Gorilla Troops Optimizer for Electrically Based Single and Double-Diode Models of Solar Photovoltaic Systems. Sustainability, 13(16), 9459. https://doi.org/10.3390/su13169459

Gunduz, M., & Aslan, M. (2021). DJAYA: A discrete Jaya algorithm for solving traveling salesman problem. Applied Soft Computing, 105, 107275. https://doi.org/10.1016/j.asoc.2021.107275

Kaveh, A., & Dadras, A. (2017). A novel meta-heuristic optimization algorithm: Thermal exchange optimization. Advances in Engineering Software, 110, 69–84.

Kennedy, J., & Eberhart, R. (n.d.). Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1987). Optimization by Simulated Annealing. In Readings in Computer Vision (pp. 606–615). Elsevier. https://doi.org/10.1016/B978-0-08-051581-6.50059-3

Koza, J. R., & Poli, R. (2005). Genetic Programming. In E. K. Burke & G. Kendall (Eds.), Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques (pp. 127–164). Springer US. https://doi.org/10.1007/0-387-28356-0_5

Kumar, A., Vohra, M., Pant, S., & Singh, S. K. (2021). Optimization techniques for petroleum engineering: A brief review. International Journal of Modelling and Simulation, 41(5), 326–334.

Kumar, J., Kumar Singh, A., Mohan, A., & Buyya, R. (2021). Metaheuristic Optimization Algorithms. In Machine Learning for Cloud Management (pp. 59–74). Chapman and Hall/CRC. https://doi.org/10.1201/9781003110101-4

Lee, K. S., & Geem, Z. W. (2004). A new structural optimization method based on the harmony search algorithm. Computers & Structures, 82(9–10), 781–798.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2010). Iterated Local Search: Framework and Applications (pp. 363–397). https://doi.org/10.1007/978-1-4419-1665-5_12

Medjahed, S. A., Ait Saadi, T., Benyettou, A., & Ouali, M. (2016). Gray Wolf Optimizer for hyperspectral band selection. Applied Soft Computing, 40, 178–186.

Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Advances in Engineering Software, 114, 163–191.

Mirjalili, S. Z., Mirjalili, S., Saremi, S., Faris, H., & Aljarah, I. (2018). Grasshopper optimization algorithm for multi-objective optimization problems. Applied Intelligence, 48(4), 805–820.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. Computers & Operations Research, 24(11), 1097–1100.

Mzili, T., Riffi, M. E., Mzili, I., & Dhiman, G. (2022). A novel discrete Rat swarm optimization (DRSO) algorithm for solving the traveling salesman problem. Decision Making: Applications in Management and Engineering, 5(2), 287–299.

Opara, K. R., & Arabas, J. (2019). Differential Evolution: A survey of theoretical analyses. Swarm and Evolutionary Computation, 44, 546–558.

Osaba, E., Yang, X.-S., & Del Ser, J. (2020). Traveling salesman problem: a perspective review of recent research and new results with bio-inspired metaheuristics. In Nature-Inspired Computation and Swarm Intelligence (pp. 135–164). Elsevier. https://doi.org/10.1016/B978-0-12-819714-1.00020-8

Pereira, J. L. J., Francisco, M. B., Diniz, C. A., Antônio Oliver, G., Cunha, S. S., & Gomes, G. F. (2021). Lichtenberg algorithm: A novel hybrid physics-based meta-heuristic for global optimization. Expert Systems with Applications, 170, 114522. https://doi.org/10.1016/j.eswa.2020.114522

Peres, F., & Castelli, M. (2021). Combinatorial Optimization Problems and Metaheuristics: Review, Challenges, Design, and Development. Applied Sciences, 11(14), 6449. https://doi.org/10.3390/app11146449

Prajapati, V. K., Jain, M., & Chouhan, L. (2020). Tabu Search Algorithm (TSA): A Comprehensive Survey. 2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE), 1–8. https://doi.org/10.1109/ICETCE48199.2020.9091743

Rahman, Md. A., & Parvez, H. (2021). Repetitive Nearest Neighbor Based Simulated Annealing Search Optimization Algorithm for Traveling Salesman Problem. OALib, 08(06), 1–17. https://doi.org/10.4236/oalib.1107520

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A Gravitational Search Algorithm. Information Sciences, 179(13), 2232–2248.

Rawat, S. S., Pant, S., Kumar, A., Ram, M., Sharma, H. K., & Kumar, A. (2022). A State-of-the-Art Survey on Analytical Hierarchy Process Applications in Sustainable Development. International Journal of Mathematical, Engineering and Management Sciences, 7(6), 883–917.

Saji, Y., & Riffi, M. E. (2016). A novel discrete bat algorithm for solving the travelling salesman problem. Neural Computing and Applications, 27(7), 1853–1866.

Simon, D. (2008). Biogeography-Based Optimization. IEEE Transactions on Evolutionary Computation, 12(6), 702–713.

Slowik, A., & Kwasnicka, H. (2020). Evolutionary algorithms and their applications to engineering problems. Neural Computing and Applications, 32(16), 12363–12379.

Sun, L. (2015). Genetic Algorithm for TSP Problem. https://doi.org/10.2991/iiicec-15.2015.319

Tanaev, V. S., Gordon, V. S., & Shafransky, Y. M. (1994). NP-Hard Problems. In Scheduling Theory. Single-Stage Systems (pp. 253–311). Springer Netherlands. https://doi.org/10.1007/978-94-011-1190-4_5

Uniyal, N., Pant, S., Kumar, A., & Pant, P. (2022). Nature-inspired metaheuristic algorithms for optimization. In Meta-heuristic Optimization Techniques (pp. 1–10). De Gruyter. https://doi.org/10.1515/9783110716214-001

Wu, C., Fu, X., Pei, J., & Dong, Z. (2021). A Novel Sparrow Search Algorithm for the Traveling Salesman Problem. IEEE Access, 9, 153456–153471.

Xu, G.-H., Zhang, T.-W., & Lai, Q. (2022). A new firefly algorithm with mean condition partial attraction. Applied Intelligence, 52(4), 4418–4431.

Yang, X.-S. (2009a). Firefly Algorithms for Multimodal Optimization (pp. 169–178). https://doi.org/10.1007/978-3-642-04944-6_14

Yang, X.-S. (2009b). Firefly Algorithms for Multimodal Optimization (pp. 169–178). https://doi.org/10.1007/978-3-642-04944-6_14

Yang, X.-S. (2010). A New Metaheuristic Bat-Inspired Algorithm (pp. 65–74). https://doi.org/10.1007/978-3-642-12538-6_6

Yang, X.-S., & Deb, S. (2009). Cuckoo Search via Lévy flights. 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), 210–214. https://doi.org/10.1109/NABIC.2009.5393690

Zhong, X. (2021). On the approximation ratio of the 3-Opt algorithm for the (1,2)-TSP. Operations Research Letters, 49(4), 515–521.