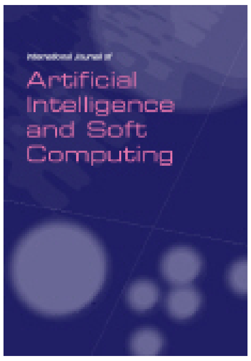


International Journal of Artificial Intelligence and Soft Computing

ISSN online: 1755-4969 - ISSN print: 1755-4950
https://www.inderscience.com/ijaisc

Genetic whale optimisation algorithm for solving travelling salesman problem

Amit Kumar

DOI: 10.1504/IJAISC.2024.10064270

Article History:

Received:	03 April 2023
Last revised:	23 January 2024
Accepted:	02 April 2024
Published online:	04 July 2024

Genetic whale optimisation algorithm for solving travelling salesman problem

Amit Kumar

Department of Information Technology,
Rajkiya Engineering College,
Ambedkar Nagar, Uttar Pradesh, 224122, India
Email: er.kumaramit2009@gmail.com

Abstract: Travelling salesman problem (TSP) is a hard combinatorial optimisation problem that has an enormous discrete search space with an excess of potential solutions. In this condition, it is impossible to carry out an exhaustive search using merely brute force. Whale optimisation algorithm (WOA) is a recent nature-inspired metaheuristic algorithm that is widely being utilised for the modern intelligent solution approach for hard optimisation problems. It is inspired by the spiral bubble-net hunting strategy of humpback whales. In this paper, a new discrete genetic operators-based whale optimisation algorithm (GWOA) has been presented for addressing the TSP. Further, experiments-based comparison of the GWOA with some recently proposed discrete particle swarm optimisation algorithms shows that the former is able to find better quality tours for TSP.

Keywords: travelling salesman problem; TSP; nature-inspired metaheuristic algorithms; whale optimisation algorithm; WOA; particle swarm optimisation; PSO.

Reference to this paper should be made as follows: Kumar, A. (2024) 'Genetic whale optimisation algorithm for solving travelling salesman problem', *Int. J. Artificial Intelligence and Soft Computing*, Vol. 8, No. 2, pp.109–128.

Biographical notes: Amit Kumar is an Assistant Professor in the Department of Information Technology at Rajkiya Engineering College Ambedkar Nagar, Uttar Pradesh, India. He earned his PhD in Computer and Systems Sciences from Jawaharlal Nehru University, New Delhi, India in 2021. His current research focuses on swarm intelligence and bio-inspired algorithms

1 Introduction

Combinatorial optimisation solves discrete optimisation problems by determining the best possible solution from a finite set of possibilities. Any combinatorial optimisation problem seeks to find the maxima (or minima) of an objective function. TSP is a well-known NP-hard combinatorial optimisation problem that is significant in operations research and theoretical computer science (Papadimitriou, 1977). It may be formulated as the problem of determining the shortest closed tour in a given network that covers every node. Even in its most basic version, TSP has several uses, including planning, logistics, and the production of microchips. It shows up in numerous domains, such as DNA sequencing, in a slightly modified form as a sub-problem. A common feature of hard combinatorial optimisation problems like TSP is large discrete search spaces with an excess of possible solutions. There is no way to conduct an exhaustive search using merely brute force under these circumstances. Thus, randomised algorithms are required. Random search algorithms and the word metaheuristic are sometimes used interchangeably. Metaheuristics are stochastic algorithms that integrate deterministic and stochastic elements and may be used to solve a variety of optimisation problems with just minor adjustments (Bianchi et al., 2009). They find optimal solutions within an acceptable time by iteratively trying to improve a candidate solution among very large spaces of candidate solutions with regard to a given measure of quality. The inspiration for them is frequently taken from the natural world. In recent years, nature-inspired metaheuristic algorithms have received widespread attention in research and these are being utilised for the modern intelligent solution approach.

In nature, a large number of insects and other small organisms are generally organised in hierarchies, e.g., ants, bees and fish, etc. These natural intelligent swarms solve many problems in nature like finding sources, division of labour among nest mates, building nests etc. For example, ants find the shortest path between their nest and the food source even if subjected to the varying environment or despite the failure of individual ants via stigmergy (Theraulaz and Bonabeau, 1999). It is a form of indirect communication mediated by change in the environment. While searching for food ants leave behind a substance called pheromone which attracts other ants. Similarly, another example is bees looking for a new location for their beehive. Communication among bees regarding the characteristics of food sources takes place in the dancing area. This dance is called a waggle dance (Tereshko and Loengarov, 2005). At the outset, numerous scouts scope potential locations. Subsequently, upon returning, they perform a distinctive dance that expresses in code the direction of the new found site. The intensity of the bee’s dance reveals the excitement for the particular location. When several scouts select the same location, the entire swarm moves. The problems that the natural intelligent swarms solve in nature have significant counterparts in numerous engineering fields of the real world. Biologists and natural scientists are constantly simulating the behaviour of natural swarms into novel numerical optimisation techniques. As a result, swarm intelligence is transpiring all the time more important research area for researchers from engineering, economics, bioinformatics, operation research and many other disciplines. By mimicking nature-inspired swarming behaviour in computing methodologies, techniques emerge for hard optimisation problems that are robust, scalable and easily distributed. A list of various swarm intelligence-inspired algorithms is shown in Table 1.

Population-based meta-heuristic algorithms exhibit a common characteristic irrespective of their nature. They split the search process into two phases: exploration and exploitation (Hu et al., 2019). The optimiser should embrace operators to globally explore the search space: during this part, movements (i.e., perturbation of style variables) ought to be randomised as much as possible. The exploitation part follows the exploration part and might be outlined as the method of investigating the promising area(s) of the search area. Therefore, exploitation pertains to the native search capability within the promising regions of search space found within the exploration part. Finding a correct balance between exploration and exploitation is the most difficult task in any meta-heuristic algorithmic rule due to the random nature of the optimisation method.

Table 1 Different swarm intelligence inspired algorithms

Name of the algorithm	*Based on technique*
Ant colony optimisation (Dorigo et al., 1991)	Foraging behaviour of ants.
Particle swarm optimisation (Kennedy and Eberhart, 1995)	Intelligent, experience-sharing, social flocking behaviour of birds
Bee system (Sato and Hagiwara, 1997)	Bees' foraging principles
Honey bee algorithm (Nakrani and Tovey, 2003)	
BeeHive (Wedde et al., 2004)	
Bee colony optimisation (Teodorović and Dell'Orco, 2005)	
Bees algorithm (Pham et al., 2005)	
Artificial bee colony (Karaboga, 2005)	
Bees swarm optimisation (Drias et al., 2005)	
Honey bee foraging (Baig and Rashid, 2007)	
Fish swarm algorithm (Li et al., 2002)	Fish school
Glow-worm swarm optimisation (Krishnanand and Ghose, 2005)	Behaviour of insects that are called glowworms
Bacterial foraging optimisation (Passino, 2002)	Group foraging behaviour of bacteria such as E. coli and M. xanthus
Cat swarm optimisation (Chu et al., 2006)	Two major behavioural traits of cats. These are termed 'seeking mode' and 'tracing mode'.
Monkey search (Mucherino and Seref, 2007)	Monkey's trees climbing behaviour
Firefly algorithm (Yang, 2008)	Flashing behaviour of fireflies
Cuckoo search (Yang and Deb, 2009)	Brooding behaviour of some cuckoo species
Mosquito host-seeking algorithm (Feng et al., 2009)	Host-seeking behaviour of mosquitoes
Bumble bee mating optimisation (Marinakis et al., 2010)	Inspired by the mating behaviour of the bumble bees
Bat algorithm (Yang, 2010)	Echolocation behaviour of micro bats
Eagle strategy (Yang and Deb, 2010)	Eagle's hunting strategy: random search by Lévy flight and intensive chase by locking its aim on the target
Krill herd algorithm (Gandomi and Alavi, 2012	Herding behaviour of krill individuals
Wolf search algorithm (Tang et al., 2012)	Imitates the way wolves search for food and survive by avoiding their enemies
Social spider optimisation (Cuevas et al., 2013)	Foraging strategy of social spiders, utilising the vibrations on the spider web to determine the positions of preys
Grey wolf optimiser (Mirjalili et al., 2014)	It mimics the leadership hierarchy and hunting mechanism of grey wolves (*Canis lupus*)in nature

Table 1 Different swarm intelligence inspired algorithms (continued)

Name of the algorithm	*Based on technique*
Elephant herding optimisation (Wang et al., 2015)	It is inspired by the herding behaviour of elephant group
Raven roosting optimisation algorithm (Brabazon et al., 2016)	Social roosting and foraging behaviour of one species of bird, the common raven
Lion optimisation algorithm (Yazdani and Jolai, 2016)	Based on simulation of the solitary and cooperative behaviours of lions such as prey capturing, mating, territorial marking, defence and the other behaviours
Whale optimisation algorithm (Mirjalili and Lewis, 2016)	It mimics the social behaviour of humpback whales. The algorithm is inspired by the bubble-net hunting strategy
Grasshopper optimisation algorithm (Saremi et al., 2017)	It mimics the behaviour of grasshopper swarms in nature for solving optimisation problems.
Salp swarm algorithm (Mirjalili et al., 2017)	The swarming behaviour of salps when navigating and foraging in oceans

Swarm intelligence algorithms, drawing inspiration from the cooperative behaviours observed in social insects and other natural systems, offer a powerful approach to address intricate optimisation challenges, including the renowned travelling salesman problem (TSP) (Wu and Duan, 2023; Tohid and Özlem, 2022; Panwar and Deep, 2023; Özer et al., 2022; Mzili et al., 2023; Gong et al., 2023).

Researchers routinely tailor the parameters and strategies of these algorithms to align with the distinctive features of the TSP. Demonstrating notable success, these algorithms have proven effective in discovering solutions that approach optimality for diverse instances of the TSP.

Whale optimisation algorithm (WOA) is a swarm-based metaheuristic algorithm (Mirjalili and Lewis, 2016). WOA copies the pattern followed by the humpback whale during hunting. It is inspired by the spiral bubble-net hunting strategy. In WOA algorithm, the hunting behaviour of whale to chase the prey and using a spiral way for bubble-net attacking mechanism is simulated. In this paper, we study how to solve TSP by WOA, and propose a new discrete WOA for discrete optimisation problems based on genetic operators of genetic algorithm. This paper is structured as follows for the remaining sections: original WOA is briefly described in Section 2 while the suggested discrete genetic operators-based whale optimisation algorithm (GWOA) for addressing TSP is presented in section 3. Results acquired using GWOA are covered in Section 4. The article is concluded in Section 5 by describing its potential future

2 Whale optimisation algorithm

WOA was proposed by Mirjalili and Lewis in 2016. This algorithm is a simulation model of the whale's unique characteristics like territorial defence, territorial takeover, laggardness exploitation and pride. From a theoretical perspective, WOA is often thought of as a worldwide optimiser as it includes exploration/exploitation ability (Mirjalili and Lewis, 2016). Furthermore, the projected hypercube mechanism defines a search space within the neighbourhood of the most effective solution and permits different search agents to take advantage of this best record within that domain. Adaptive variation of the search vector *Q* allows the WOA algorithm to smoothly transit between exploration and exploitation: by decreasing Q, some iterations are devoted to exploration ($|Q| \geq 1$) and the rest is dedicated to exploitation ($|Q| < 1$). Remarkably, WOA includes only two main internal parameters to be adjusted (*Q* and *P*). Three main concepts of the whale optimisation algorithm are encircling prey, bubble net attacking and the search for prey. These are discussed below:

2.1 Encircling prey

Humpback whales acknowledge the position of the prey and then encircle them (Mirjalili and Lewis, 2016). Since the position of the best solution within the search space is not evident a priori, WOA rule assumes that the best candidate solution is the target prey or is near to the optimum. After the simplest search agent is outlined, the opposite search agents can therefore try and update their positions towards the simplest search agent. This behaviour is modelled by the subsequent equations.

$$\vec{L} = \left| \vec{P} \cdot \vec{Y}^*(t) - \vec{Y}(t) \right| \tag{1}$$

$$\vec{Y}(t+1) = \vec{Y}^*(t) - \vec{Q} \cdot \vec{L} \tag{2}$$

where *t* indicates the current iteration, *Q* and *P* are coefficient vectors, Y^* is the position vector of the best solution obtained so far, *Y* is the position vector, | | is the absolute value, and · is an element-by-element multiplication. It is worth mentioning here that Y^* should be updated in each iteration if there is a better solution.

The vectors *Q* and *P* are calculated as follows:

$$\vec{Q} = 2\vec{m} \cdot \vec{r}_1 - \vec{m} \tag{3}$$

$$\vec{P} = 2 \cdot \vec{r}_2 \tag{4}$$

where *a* is linearly decreased from 2 to 0 over the course of iterations (in both exploration and exploitation phases) and *r* is a random vector in [0, 1].

The same idea is extended to a search space with *n* dimensions and therefore the search agents can move in hyper-cubes around the solution obtained to this point. As mentioned in the previous section, the humpback whales conjointly attack the prey with the bubble-net strategy.

2.2 Bubble-net attacking

The bubble-net attacking behaviour of humpback whales can be described by two techniques as follows (Mirjalili and Lewis, 2016):

2.2.1 Shrinking encircling mechanism

This behaviour is achieved by decreasing the value of *m* in equation (3). Note that the fluctuation range of *Q* is also decreased by *m*. *Q* is a random value in the interval [–*m*, *m*] where *a* is decreased from 2 to 0 over the course of different iterations. Using random values for Q in [–1, 1], the new position of a search agent can be defined anywhere in between the original position of the search agent and the position of the current best search agent.

2.2.2 Spiral updating position

This approach measures the distance between the whale located at (Y, Z) and the prey located at (Y^*, Z^*). A spiral equation is made between the position of whale and prey to copy the helix-shaped movement of humpback whales as follows:

$$\vec{Y}(t+1) = \vec{L}' \cdot e^{bl} \cdot \cos(2\pi l) + \vec{Y}^*(t) \tag{5}$$

where $\vec{L}' = |\vec{Y}^*(t) - \vec{Y}(t)|$ and indicates the distance of the i^{th} whale to the prey (best solution obtained so far), b is a constant for defining the shape of the logarithmic spiral, l is a random number in [–1, 1], and · is an element-by-element multiplication.

The humpback whales swim round the prey along a shrinking circle and on a spiral shaped path at the same time. To model this coincidental behaviour, it is assumed that there is an equal likelihood of deciding between either the shrinking peripheral mechanism or the spiral model to update the position of whales throughout optimisation. The mathematical model is as follows:

$$\vec{Y}(t+1) = \begin{cases} \vec{Y}^*(t) - \vec{Q} \cdot \vec{L} & \text{if } p < 0.5 \\ \vec{L}' \cdot e^{bl} \cdot \cos(2\pi l) + \vec{Y}^*(t) & \text{it } p \geq 0.5 \end{cases} \tag{6}$$

where p is a random number in [0, 1]. In WOA, the humpback whales find prey randomly. The mathematical model of the search is as follows.

2.3 Search for prey

WOA uses Q with random values greater than 1 or less than –1 to force the search agents to go far away from a reference whale. $|Q|> 1$ emphasises exploration and allows the WOA algorithmic program to perform a worldwide search. The mathematical model is as follows:

$$\vec{L} = \theta \left| \vec{P} \cdot \vec{Y}_{rand} - \vec{Y} \right| \tag{7}$$

$$\vec{Y}(t+1) = \vec{Y}_{rand} - \vec{Q} \cdot \vec{L} \tag{8}$$

where Y_{rand} is a random position vector (random whale) chosen from the current population of the whales.

WOA cannot be utilised to directly resolve discrete optimisation problems because it was developed for continuous optimisation where it can converge quickly on the best solution or solutions in the high dimension. TSP, however, is an NP-hard discrete problem. Since the variables are integers, the arrangement of their values in the various dimensions indicates a distinct solution with a different objective function value. As a result, in order to solve TSP efficiently, the typical whale optimisation technique must be adjusted to account for the unique requirements of the issue. Researchers have suggested many DWOA algorithm variations in the literature (Zamani and Nadimi-Shahraki, 2016; Abdel-Basset et al., 2017; Mafarja and Mirjalili, 2017; Hussien et al., 2018; Reddy et al., 2018; Hussien et al., 2019). In Zhang et al. (2021), a discrete whale optimisation algorithm with variable neighbourhood search was proposed to address the TSP. In this paper a new discrete GWOA is presented to address TSP, which is discussed next.

3 GWOA for TSP

TSP is a well-known NP-hard problem in optimisation in the disciplines of computer science and operations research. It asserts that given a set of cities and distances between each pair of cities, a salesperson must visit each city precisely once and return to the city where he began his tour. What is the shortest possible path that the salesman must take to finish his tour? The solution to the problem can be found out by comparing each round-trip route to find the shortest one. But with the growing number of cities, the number of possible roundtrips outpaces the capacity of even the most powerful computers. Therefore, classical mathematical methods can not be used to address the TSP. A wide variety of alternative solutions have also been developed in an effort to discover the optimal solution as quickly as feasible. Nature-inspired algorithms have been successful in finding the solutions that are very near to the optimal. In this paper, TSP is solved using GWOA algorithm and its performance is compared with discrete particle swarm optimisation (PSO) algorithms on the basis of tour length.

Suppose there are n cities and the sequence of cities that shall be visited serves as the coding for the solution. Since each element in the solution is represented by a city number, any solution X can be represented as $(x_1, x_2, x_3, \ldots, x_n)$, where $x_j \in [1, n]$ and any $x_i \neq x_j$. The following fitness function is used to calculate the tour length for the TSP:

$$f = \sum_{i=1}^{n-1} d\left(x_i, x_{i+1}\right) + d\left(x_n, x_1\right) \tag{9}$$

where $d(x_i, x_{i+1})$ is the distance between the cities x_i and x_{i+1} · $d(x_n, x_1)$ is the distance between the city x_n and the city x_1.

3.1 Proposed GWOA

In ElMousel et al. (2021), a novel discrete WOA was proposed for solving the capacitated vehicle routing problem. In this paper this discrete WOA has been modified to apply it on TSP. Similar to humpback whales hunting in groups, GWOA starts with a random population of TSP solutions as whales. The following three steps make up the hunting procedure.

1 *Encircling the target prey:* during hunting, humpback whales explore the ocean to identify the whereabouts of their prey. As soon as they locate the prey, they start enveloping them. Since, we are solving TSP, we are looking for some solution that will be the shortest route feasible among others. In this case, the search space which consists of all feasible TSP solutions, corresponds to the ocean. Each solution of the TSP is representing a unique whale agent. Furthermore, it is believed that the current best TSP solution is near the target prey, or the best solution, as the location of the ideal prey is not known beforehand. All the other whales update their positions in accordance with the best whale agent.

2 *Bubble-net attacking:* this is the exploitation phase of WOA where humpback whales exhibit two possible types of movements: either travelling in a narrowing circle around their target prey or travelling in a spiral path. In the first case, each whale's new position can be specified anywhere between its initial position and the position of the best whale at the time. The second choice, however, updates the spiral location in accordance with how far apart each whale is from its target prey.

3 *Searching for prey:* this is the exploitation phase of WOA where the position of a search agent is updated based on a randomly selected search agent rather than using the best search agent found thus far.

The objective function of the TSP, which is shortest possible tour length that the salesman must cover to visit each city precisely once and return to the city where he began his tour, is used to determine each whale's fitness.

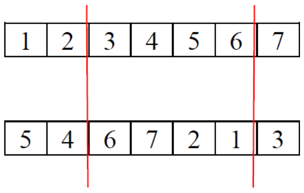
WOA was originally proposed for continuous optimisation that allows the updating of whales' positions in a continuous domain. TSP is a combinatorial optimisation problem that can not be addressed by the original WOA. So, a discrete WOA that updates the position of whales on the basis of the crossover operator of the genetic algorithm is presented. Two different crossover operators: partially mapped crossover and order crossover operators are used which are described next.

3.1.1 *Partially mapped crossover (PMX) operator*

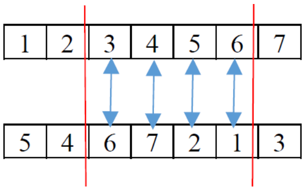
Step 1 Choose two parents to mate.

P:	1	2	3	4	5	6	7		$P2$:	5	4	6	7	2	1	3

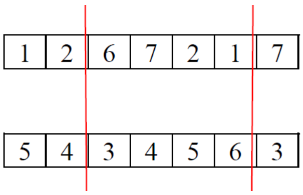
Step 2 Using crossover points, choose a substring at random from the parents.



Step 3 Implement the two-point crossover.



Step 4 Identify mapping relationships using substrings



$1 \leftrightarrow 6 \leftrightarrow 3$

$7 \leftrightarrow 4$

$2 \leftrightarrow 5$

Step 5 To legitimise offsprings in unselected substrings, use the mapping.

$O1:$

3	5	6	7	2	1	4

$O2:$

2	7	3	4	5	6	1

3.1.2 Ordered crossover operator

It is a variation of PMX with a different repairing procedure. It creates offsprings by picking a parent's subtour and keeping the relative order of the other parent's bits.

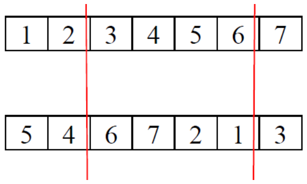
$P1:$

1	2	3	4	5	6	7

$P2:$

5	4	6	7	2	1	3

Step 1 Select a substring from a parent at random.



Step 2 Copy the substring into the corresponding location of it to create a proto-child.

$O1:$

		3	4	5	6	

$O2:$

		6	7	2	1	

Step 3 Remove the cities from the second parent that are already in the substring. The cities that the proto-child requires are included in the resulting sequence of cities.

Step 4 In order to make offspring, place the cities in the unfixed positions of the proto-child from left to right in the prescribed order of the sequence.

$O1:$

2	1	3	4	5	6	7

$O2:$

4	5	6	7	2	1	3

The exploitation phase: this phase achieves the local minimum by updating the whales' positions using either the spiral model or the declining encircling process with an equal probability.

$$X(t+1) = \begin{cases} f_1\left(X(t), X_{best}(t)\right), & \text{if } p' < 0.5 \\ f_2\left(X(t), X_{best}(t)\right), & \text{if } p' \geq 0.5 \end{cases} \tag{10}$$

where f_1 and f_2 are partially mapped crossover operator and order crossover operators respectively between the current whale and the best whale. p' is a random number between 0 and 1. The fitter of the two offspring produced by the crossover operator is chosen as the updated solution.

The exploration phase: this phase improves the ability of the global search by choosing a random whale from the population and applying a crossover operator between this random whale and the current whale. Again, the fitter of the two offspring produced by the crossover operator is chosen as the updated solution.

$$X(t+1) = f_3\left(X(t), X_{rand}(t)\right) \tag{11}$$

where f_3 represents the ordered crossover operator between the randomly chosen whale and the current whale.

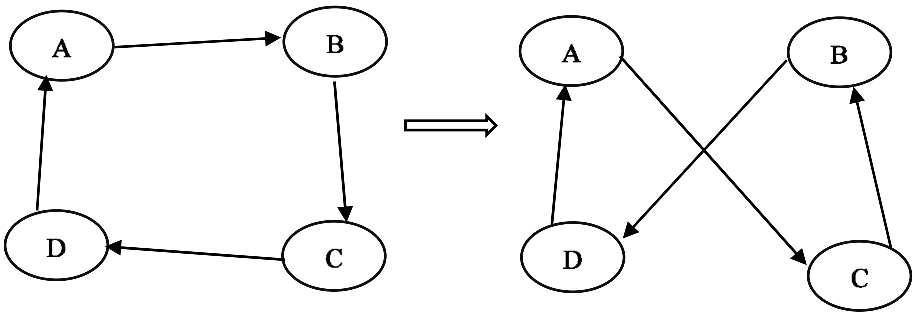
Balancing probability: GWOA strikes a balance between the exploration phase and the exploitation phase by comparing the balancing probability (p_b) with a random number $r \in (0, 1)$. If $r \leq p_b$, the exploitation phase is used to update the solution, otherwise, the solution is updated by using the exploration phase. Balancing probability is calculated as follows:

$$p_b = 1 - \left(\frac{iter}{iter_{\max}} \right)^2 \tag{12}$$

where *iter* is the current iteration and $iter_{\max}$ is the total number of iteration.

2-opt mutation: 2-opt mutation (Croes, 1958) is a local search strategy for TSP to precise solutions quickly. In the proposed GWOA, the concept of 2-opt mutation is used so as to modify the original WOA for TSP. 2-opt mutation is applied on the best whale in each iteration.

Figure 1 The routes for TSP before and after 2-opt



The pseudo-code of the GWOA is shown in Figure 2.

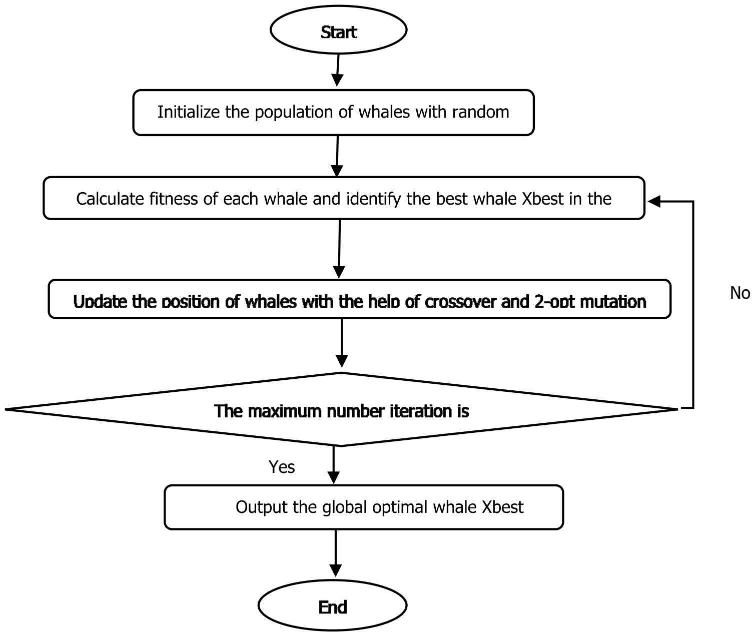
Initially, the number of whale search agent in the population are defined as *ps* and a termination criterion is set as the maximum number of iterations. A random tour is assigned to each of the whale in the population. Then, fitness value (i.e., tour length) of each whale search agent is computed. The best whale X_{best} in the population is determined. In each iteration step, the position $X(t)$of each whale is updated based on the position of the best whale using the crossover and 2-opt mutation operators discussed above. The fitness value of each new whale is computed. The best whale search agent is updated if possible. Finally, the best whale X_{best} is returned as the outcome of the whole optimisation procedure.

The flow chart of GWOA algorithm is also shown in Figure 3.

Figure 2 Pseudo-code of the GWOA

```
Input: $ps$: Population size, $I_M$: Maximum number of iterations
Output: The best whale $X_{best}$
Initialize whales with random tours
Calculate fitness of each whale and identify the best whale $X_{best}$ in the population
while($t < I_M$) do
    for each whale in population $ps$ do
        if($p' < 0.5$) then
            if($rand > p_b$) then
                partially mapped crossover ($X(t)$, $X_{best}(t)$)
            else
                ordered crossover operator ($X(t)$, $X_{rand}(t)$)
            end
        else
            ordered crossover operator ($X(t)$, $X_{best}(t)$)
        end
        2-opt mutation on the best whale $X_{best}$
    end
end
```

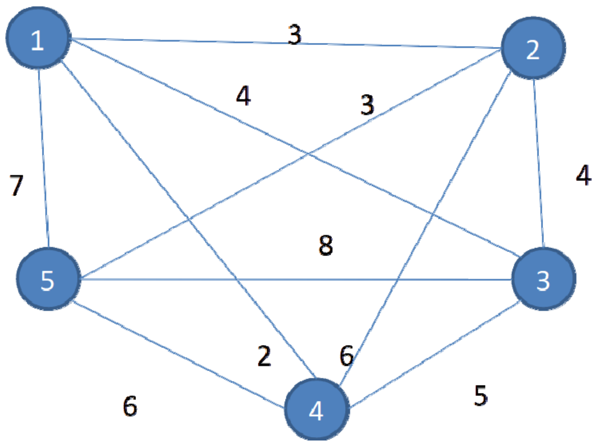
Figure 3 Flowchart of GWOA


Start
Initialize the population of whales with random
Calculate fitness of each whale and identify the best whale Xbest in the
Update the position of whales with the help of crossover and 2-opt mutation
The maximum number iteration is
No
Yes
Output the global optimal whale Xbest
End

4 Illustrative example: tackling TSP with GWOA

Consider the fully connected graph of five vertices as depicted in Figure 4.

Figure 4 Fully connected graph with five vertices (see online version for colours)



In TSP, the aim is to find the shortest Hamiltonian cycle in a fully connected graph. In this example, the number of different Hamiltonian cycles is (5 – 1)!/2 = 12. The possible tours and respective costs are shown in Table 2.

Table 2 Possible TSP tours and respective costs

Sr. no.	*Tour*	*Cost*
1	(1, 2, 3, 4, 5)	25
2	(1, 5, 3, 4, 2)	29
3	(1, 4, 3, 2, 5)	21
4	(1, 3, 2, 4, 5)	27
5	(1, 2, 5, 4, 3)	21
6	(1, 3, 5, 2, 4)	23
7	(1, 4, 3, 5, 2)	21
8	(1, 2, 3, 5, 4)	23
9	(1, 3, 5, 4, 2)	27
10	(1, 3, 4, 2, 5)	25
11	(1, 4, 5, 2, 3)	19
12	(1, 4, 2, 3, 5)	27

Let the population size *ps* = 6. A population of six whales is initialised randomly from the available fully connected graph shown in Table 3.

Table 3 Initial population of six whales

Whale (i)	*Position (X_i)*
1	(1, 5, 3, 4, 2)
2	(1, 3, 2, 4, 5)
3	(1, 3, 5, 2, 4)
4	(1, 4, 3, 2, 5)
5	(1, 3, 4, 2, 5)
6	(1, 4, 5, 2, 3)

The fitness, i.e., tour length of each of the six whales is computed, as shown in Table 4.

Table 4 Fitness or tour length of six whales

Whale (i)	*Position (X_i)*	*Fitness (tour length)*
1	(1, 5, 3, 4, 2)	29
2	(1, 3, 2, 4, 5)	27
3	(1, 3, 5, 2, 4)	23
4	(1, 4, 3, 2, 5)	21
5	(1, 3, 4, 2, 5)	25
6	(1, 4, 5, 2, 3)	19

The best whale in the population with the shortest possible tour length is X_{best} = (1, 4, 5, 2, 3).

Now, for each whale of the population in position $X_i(t)$ the next position $X_i(t + 1)$ is computed.

The first iteration of position update of the first whale is illustrated below:

$$\text{Current iteration counter } t = 1,\ X_1(1) = (1, 5, 3, 4, 2),\ X_{best}(1) = (1, 4, 5, 2, 3)$$

Let the maximum number of iterations $IM = 100$.

So balancing probability:

$$\begin{aligned} p_b &= 1 - \left(\frac{iter}{iter_{\max}}\right)^2 \\ &= 1 - \left(\frac{1}{100}\right)^2 \\ &= 1 - .0001 \\ p_b &= 0.9999 \end{aligned}$$

Let $p' = 0.7532$ and $rand = 0.4324$

Now $p' > 0.5$, so ordered crossover operator is performed between $X_1(1)$ = (1, 5, 3, 4, 2) and $X_{best}(1)$ = (1, 4, 5, 2, 3).

$P1$: | 1 | 5 | 3 | 4 | 2 | $P2$: | 1 | 4 | 5 | 2 | 3 |

Random substrings are selected from the two parents as follows:

1	5	3	4	2
1	4	5	2	3

These substrings are copied into the corresponding locations to create two proto-children as follows.

$O1:$

	5	3		

$O2:$

	4	5		

Following this, the bits from the opposite parent are duplicated in the same sequence but without the existing bits, beginning at the second cut point of one parent.

For example, the bit sequence in the second parent from the 2nd cut point is '2 →3→ 1→ 4→ 5'. After removing the existing bits 5 and 3 of the first offspring the new sequence is '2→1→4'. This sequence is duplicated in the first offspring starting from the 2nd cut point. Similarly, the other offspring is generated.

$O1:$

4	5	3	2	1

$O2:$

3	4	5	2	1

Now, one of the two offsprings is chosen randomly as the updated first whale.

So the updated first whale after the first iteration is

3	4	5	2	1

5 Experimental results

The proposed algorithm GWOA and three recent discrete PSO algorithms: standard particle swarm optimisation (SPSO) (Chuan and Quanyuan, 2007), adaptive particle swarm optimisation (APSO) (Zhan et al., 2009) and quantum-based particle swarm optimisation (QPSO) (Ho et al., 2013) were implemented using MATLAB 2015a in windows 10 environment. These algorithms were compared by conducting experiments on an Intel-based 2.5 GHz PC having 8 GB RAM. Table 5 shows the comparisons of the lengths of shortest tours found by these algorithms. The TSP problem set is taken from the http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/. Number of iterations and population size are 400 and 400 for all the test cases.

Table 5 Experimental results of solving TSP using GWOA with SPSO, QPSO and APSO.

Sr. no.	*Problem name*	*GWOA*	*SPSO*	*QPSO*	*APSO*
1	Att48	50,911.53	124,922.3	106,805.4	88,240.69
2	Bayg29	12,579.9	16,587.71	15,918.07	15,663.98
3	Burma14	31.22692	31.20877	31.78388	32.47089
4	Ulysses16	74.10874	77.22789	76.59773	75.63054
5	Ulysses22	79.10002	90.35913	90.64342	94.90065

The performance of these algorithms was also compared graphically by plotting graphs between the best fitness and iteration counter when applied to these five benchmark TSPs in Figures 5–9.

Figure 5 Fitness vs. iterations, Att48 (see online version for colours)

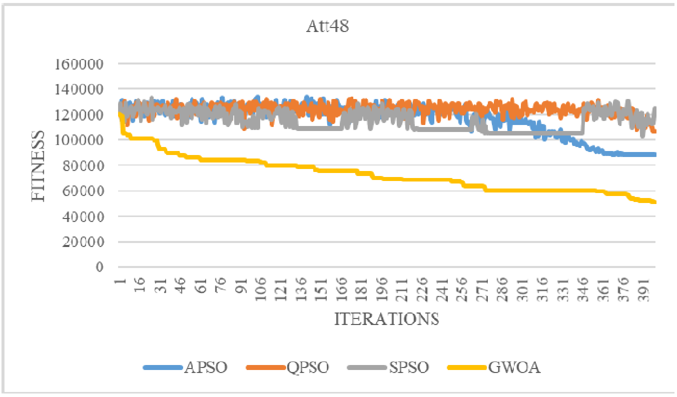


Figure 6 Fitness vs. iterations, Bayg29 (see online version for colours)

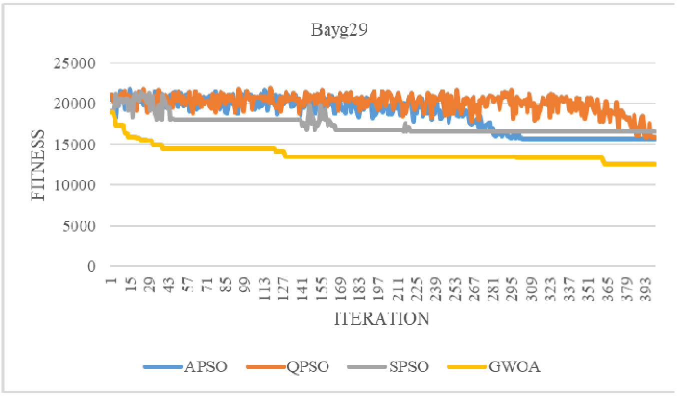


Figure 7 Fitness vs. iterations, Burma14 (see online version for colours)

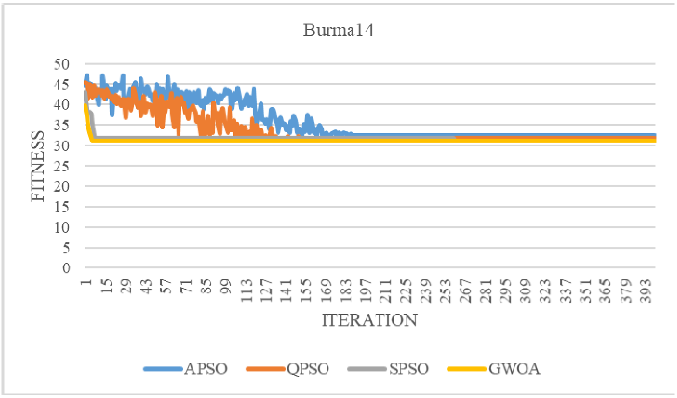


Figure 8 Fitness vs. Iterations, Ulysses16 (see online version for colours)

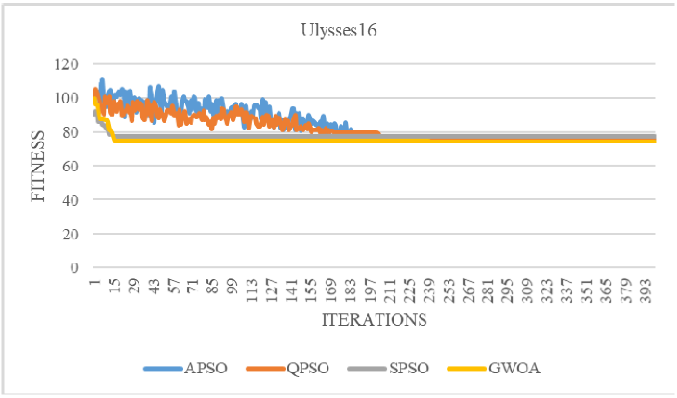
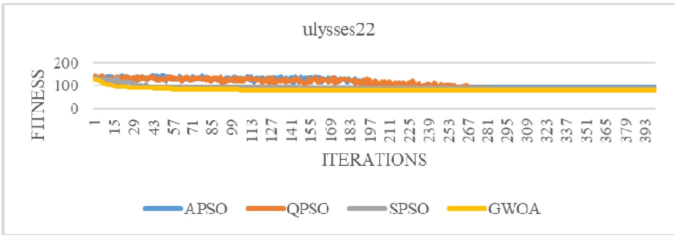


Figure 9 Fitness vs. iterations, Ulysses22 (see online version for colours)

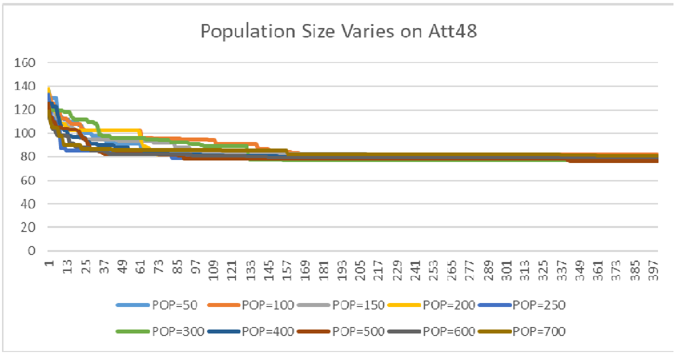


The effect of making the changes in the initial population size for GWOA algorithm on the results of Att48 problem set is shown in the Table 6 and Figure 10.

Table 6 Effect of initial population of WOA on the results of Att48 (see online version for colours)

Sr. no.	*Population size*	*Fitness result*
1	50	77.09092
2	100	76.77059
3	150	79.10002
4	200	76.12375
5	250	79.44039
6	300	80.95955
7	400	77.09092
8	500	76.77059
9	600	79.10002
10	700	76.12375

Figure 10 Population size vs. fitness, Att48 (see online version for colours)



6 Conclusions

In this paper, a new discrete WOA has been presented to solve the TSP. The WOA algorithm, which is was inspired by the way humpback whales hunt, was initially proposed to solve continuous optimisation problems. The original WOA has been hybridised by combining the approach with crossover operators from the GA and adding the 2-opt exchange to further enhance the solution. Simulations were carried out on five TSP instances using GWOA, and its performance was compared with three other state-of-the-art discrete PSO algorithms. Experimental results showed that the proposed algorithm GWOA performed well for TSP. Based on this, it can be concluded that GWOA can be used to solve other large discrete combinatorial optimisation problems.

References

Abdel-Basset, M., El-Shahat, D. and Sangaiah, A.K. (2017) 'A modified nature inspired meta-heuristic whale optimization algorithm for solving 0-1 knapsack problem', *International Journal of Machine Learning & Cybernetics*, Vol. 1, pp.1–20.

Baig, A. and Rashid, M. (2007) 'Honey bee foraging algorithm for multimodal & dynamic optimization problems', *GECCO'07: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, p.169.

Bianchi, L., Dorigo, M., Gambardella, L.M. and Gutjahr, W.J. (2009) 'A survey on metaheuristics for stochastic combinatorial optimization', *Natural Computing*, Vol. 8, No. 2, pp.239–287.

Brabazon, A., Cui, W. and Neil, M. (2016) 'The raven roosting optimization algorithm', *Soft Computing*, Vol. 20, No. 2, pp.225–245.

Chu, S.A., Tsai, P.W. and Pan, J.S. (2006) 'Cat swarm optimization', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 4099 LNAI, pp.854–858.

Chuan, L. and Quanyuan, F. (2007) 'The standard particle swarm optimization algorithm convergence analysis and parameter selection', *Third International Conference on Natural Computation (ICNC 2007)*, Haikou, pp.823–826.

Croes, G.A. (1958) 'A method for solving traveling-salesman problems', *Oper. Res.*, Vol. 6, No. 6, pp.791–812.

Cuevas, E., Cienfuegos, M., Zaldívar, D. and Pérez-Cisneros, M. (2013) ‘A swarm optimization algorithm inspired in the behavior of the social-spider’, *Expert Systems with Applications*, Vol. 40, No. 16, pp.6374–6384.

Dorigo, M., Colorni, A and Maniezzo, V. (1991) *Positive Feedback as a Search Strategy*, Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.

Drias, H., Sadeg, S. and Yahi, S. (2005) ‘Cooperative bees swarm for solving the maximum weighted satisfiability problem’, *IWAAN International Work Conference on Artificial and Natural Neural Networks*, pp.318–325.

ElMousel, A.S., Khairy, O.M., Shehata, O.M. and Morgan, E.I. (2021) ‘A novel discrete whale optimization algorithm for solving the capacitated vehicle routing problem’, *2021 7th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, DOI: 10.1109/icmre51691.2021.9384842.

Feng, X., Lau, F.C.M. and Gao, D. (2009) ‘A new bio-inspired approach to the traveling salesman problem’, *Complex Sciences, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, February, Vol. 5, pp.1310–1321, Springer Berlin Heidelberg.

Gandomi, A.H. and Alavi, A.H. (2012) ‘Krill Herd: a new bio-inspired optimization algorithm’, *Communications in Nonlinear Science and Numerical Simulation*, Vol. 17, No. 12, pp.4831–4845.

Gong, X., Rong, Z., Wang, J. et al. (2023) ‘A hybrid algorithm based on state-adaptive slime mold model and fractional-order ant system for the travelling salesman problem’, *Complex Intell. Syst.*, Vol. 9, No. 3, .3951–3970.

Ho, S.L., Yang, S., Ni, G. and Huang, J. (2013) ‘A quantum-based particle swarm optimization algorithm applied to inverse problems’, *IEEE Transactions on Magnetics*, Vol. 49, No. 5, pp.2069–2072.

Hu, Z., Jianyong, S., Tonglin, L., Ke, Z. and Qingfu, Z. (2019) ‘Balancing exploration and exploitation in multiobjective evolutionary optimization’, *Information Sciences*, , p.S0020025519304499, DOI: 10.1016/j.ins.2019.05.046.

Hussien, A.G., Hassanien, A.E., Houssein, E.H., et al. (2019) ‘S-shaped binary whale optimization algorithm for feature selection’, *Advances in Intelligent Systems and Computing*, pp.79–87, https://doi.org/10.1007/978-981-10-8863-6.

Hussien, A.G., Houssein, E.H. and Hassanien, A.E. (2018) ‘A binary whale optimization algorithm with hyperbolic tangent fitness function for feature selection’, *Eighth International Conference on Intelligent Computing and Information Systems*, IEEE, pp 166–172.

Karaboga, D. (2005) *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Technical report, Computer Engineering Department, Engineering Faculty, Erciyes University.

Kennedy, J. and Eberhart, R. (1995) ‘Particle swarm optimization’, *IEEE International Conference on Neural Networks*, Vol. 4, pp.1942–1948.

Krishnanand, K. and Ghose, D. (2005) ‘Detection of multiple source locations using a glowworm metaphor with applications to collective robotics’, *IEEE Swarm Intelligence Symposium*, Pasadena, CA, USA, pp.84–91.

Li, L.X., Shao, Z.J. and Qian, J.X. (2002) ‘An optimizing method based on autonomous animals: fish-swarm algorithm’, *Syst. Eng. Theory Practice*, Vol. 22, No. 11, pp.32–38.

Mafarja, M.M. and Mirjalili, S. (2017) ‘Hybrid whale optimization algorithm with simulated annealing for feature selection’, *Neurocomputing*, Vol. 260, pp.302–312, 342.

Marinakis, Y., Marinaki, M. and Matsatsinis, N. (2010) ‘A bumble bees mating optimization algorithm for global unconstrained optimization problems’, *NICSO 2010, SCI*, Vol. 284, pp.305–318.

Mirjalili, S. and Lewis, A. (2016) ‘The whale optimization algorithm’, *Advances in Engineering Software*, Vol. 95, pp.51–67.

Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H. and Mirjalili, S.M. (2017) 'Salp swarm algorithm: a bio-inspired optimizer for engineering design problems', *Advances in Engineering Software*, Vol. 114, pp.163–191.

Mirjalili, S., Mirjalili, S.M. and Lewis, A. (2014) 'Grey wolf optimizer', *Advances in Engineering Software*, Vol. 69, pp.46–61.

Mucherino, A. and Seref, O. (2007) 'Monkey search: a novel meta-heuristic search for global optimization, data mining, system analysis and optimization in biomedicine', in Seref, O., Kundakcioglu, O.E. and Pardalos, P.M. (Eds.): *AIP Conference Proceedings*, Vol. 953, pp.162–173.

Mzili, T., Mzili, I. and Riffi, M.E. (2023) 'Artificial rat optimization with decision-making: a bio-inspired metaheuristic algorithm for solving the traveling salesman problem', *Decision Making: Applications in Management and Engineering*, Vol. 6, No. 2, pp.150–176.

Nakrani, S. and Tovey, C. (2003) 'On honey bees and dynamic allocation in an internet server colony', *Proceedings of 2nd International Workshop on the Mathematics and Algorithms of Social Insects*.

Özer, S., Baykasoğlu, A. and Kilinçci, Ö. (2022) 'Application of chicken swarm optimization to travelling salesman problem and a reviewing of similar studies', *Journal of New Results in Engineering and Natural Sciences*, Vol. 15, pp.65–72.

Panwar, K. and Deep, K. (2023) 'Discrete salp swarm algorithm for Euclidean travelling salesman problem', *Appl. Intell.*, Vol. 53, pp.11420–11438.

Papadimitriou, C.H. (1977) 'The Euclidean travelling salesman problem is NP-complete', *Theoretical Computer Science*, Vol. 4, No. 3, pp.237–244.

Passino, K. (2002) 'Biomimicry of bacterial foraging for distributed optimization and control', *IEEE Control Systems Magazine*, Vol. 22, No. 3, pp.52–67.

Pham, D., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S. and Zaidi, M. (2005) *The Bees Algorithm*, Technical report, Cardiff University, UK.

Reddy, K.S., Panwar, L., Panigrahi, B.K. et al. (2018) 'Binary whale optimization algorithm: a new metaheuristic approach for profit-based unit commitment problems in competitive electricity markets', *Eng. Optim.*, Vol. 51, No. 3, pp.369–389.

Saremi, S., Mirjalili, S. and Lewis, A. (2017) 'Grasshopper optimisation algorithm: theory and application', *Advances in Engineering Software*, Vol. 105, pp.30–47.

Sato, T. and Hagiwara, M. (1997) 'Bee system: finding solution by a concentrated search', *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 4, No. C, pp.3954–3959.

Tang, R., Fong, S., Yang, X.S. and Deb, S. (2012) 'Wolf search algorithm with ephemeral memory', *2012 Seventh International Conference on Digital Information Management (ICDIM)*, pp.165–172.

Teodorović, D. and Dell'Orco, M. (2005) 'Bee colony optimization–a cooperative learning approach to complex transportation problems', *Advanced OR and AI Methods in Transportation: Proceedings of 16th Mini–EURO Conference and 10th Meeting of EWGT*, 13–16 September 2005, Publishing House of the Polish Operational and System Research, Poznan, pp.51–60.

Tereshko, V. and Loengarov, A. (2005) 'Collective decision making in honey-bee foraging dynamics', *Comput. Inf. Syst.*, Vol. 9, No. 3, pp.1–7.

Theraulaz, G. and Bonabeau, E. (1999) 'A brief history of Stigmergy', *Artificial Life*, Vol. 5, No. 2, pp.97–116.

Tohid, Y. and Özlem, A. (2022) 'Solving the traveling salesman problem by adding a mutation approach to a grasshopper optimization algorithm', *1st International Conference on Engineering, Natural and Social Sciences*, Turkey, Konya, Vol. 1.

Wang, G.G, Deb, S. and Coelho, L. (2015) 'Elephant herding optimization', *3rd International Symposium on Computational and Business Intelligence (ISCBI)*.

Wedde, H., Farooq, M. and Zhang, Y. (2004) 'Beehive: an efficient fault-tolerant routing algorithm inspired by honey bee behavior', in Dorigo, M. (Ed.): *Ant Colony Optimization and Swarm Intelligence*, pp.83–94, Springer, Berlin.

Wu, J. and Duan, Q. (2023) 'An algorithm for solving travelling salesman problem based on improved particle swarm optimisation and dynamic step Hopfield network', *International Journal of Vehicle Design*, Vol. 91, No. 1–3, pp.208–231.

Yang, X.S. (2008) *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, UK.

Yang, X.S. (2010) 'A new metaheuristic bat-inspired algorithm', *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pp.65–74.

Yang, X.S. and Deb, S. (2009) 'Cuckoo search via Lévy flights', *Proceedings of World Congress on Nature and Biologically Inspired Computing (NaBIC 2009)*, IEEE Publications, USA, pp.210–214.

Yang, X.S. and Deb, S. (2010) 'Eagle strategy using Lévy walk and firefly algorithms for stochastic optimization', *Nature Inspired Cooperative Strategies for Optimization (NICSO2010)*, pp.101–111, Springer.

Yazdani, M. and Jolai, F. (2016) 'Lion optimization algorithm (LOA): a nature-inspired metaheuristic algorithm', *Journal of Computational Design and Engineering*, Vol. 3, No. 1, pp.24–36.

Zamani, H. and Nadimi-Shahraki, M.H. (2016) 'Feature selection based on whale optimization algorithm for diseases diagnosis', *International Journal of Computer Science and Information Security (IJCSIS)*, Vol. 14, No. 9, pp.1243–1247.

Zhan, Z.H., Zhang, J., Li, Y. and Chung, H.S.H. (2009) 'Adaptive particle swarm optimization', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 39, No. 6, pp.1362–1381.

Zhang, J., Hong, L. and Liu, Q. (2021) 'An improved whale optimization algorithm for the traveling salesman problem', *Symmetry*, Vol. 13, No. 1, p.48, https://doi.org/10.3390/sym13010048.