

Improved discrete plant propagation algorithm for solving the traveling salesman problem

Hussein Fouad Almazini¹, Salah Mortada¹, Hassan Fouad Abbas Al-Mazini¹, Hayder Naser Khraibet AL-Behadili², Jawad Alkenani²

¹School of Computing, Universiti Utara Malaysia, Kedah, Malaysia

²Department of Computer Science, Shatt Alarab University College, Basrah, Iraq

Article Info

Article history:

Received Feb 4, 2021

Revised Oct 14, 2021

Accepted Oct 22, 2021

Keywords:

Genetic algorithm

Metaheuristic

Optimization

Plant intelligence

ABSTRACT

The primary goal of traveling salesman problem (TSP) is for a salesman to visit many cities and return to the starting city via a sequence of potential shortest paths. Subsequently, conventional algorithms are inadequate for large-scale problems; thus, metaheuristic algorithms have been proposed. A recent metaheuristic algorithm that has been implemented to solve TSP is the plant propagation algorithm (PPA), which belongs to the rose family. In this research, this existing PPA is modified to solve TSP. Although PPA is claimed to be successful, it suffers from the slow convergence problem, which significantly impedes its applicability for getting good solution. Therefore, the proposed partial-partitioned greedy algorithm (PPGA) offers crossover and three mutation operations (flip, swap, and slide), which allow local and global search and seem to be wise methods to help PPA in solving the TSP. The PPGA performance is evaluated on 10 separate datasets available in the literature and compared with the original PPA. In terms of distance, the computational results demonstrate that the PPGA outperforms the original PPA in nine datasets which assures that it is 90% better than PPA. PPGA produces good solutions when compared with other algorithms in the literature, where the average execution time reduces by 10.73%.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Hussein Fouad Almazini

School of Computing, Universiti Utara Malaysia

Sintok, Kedah, Malaysia

Email: h.almazni22@gmail.com

1. INTRODUCTION

Combinatorial optimization problems (COPs) are a subset of mathematical optimization problems that are used in different fields, regardless of whether the structure is complex or simple. Various functional problems, such as the traveling salesman problem (TSP), the assembly line balancing problem, the shortest path tree, and the minimum period tree, can be classified as COPs. TSP is commonly used to assess the efficiency of recently designed approaches to COPs and of those relevant to many significant fields, such as engineering, logistics, and transport. TSP is an NP-hard problem following a Hamiltonian cycle with minimal expense [1], [2]. In the TSP principle, the vendor begins from one city, visits all other cities precisely once upon a time, and returns to the beginning city seeking to get a closed tour with lowest expense. The tour expense depends directly on the tour length [3], [4]. Many researchers have suggested solving TSP, with its simple description and very complicated solution. Initially, exact and approximate (heuristic or metaheuristic) approaches were developed to resolve TSP [5], [6]. Exact methods can solve small TSPs optimally. By contrast, heuristic methods are preferable for large TSPs. Moreover, certain greedy, principle-based algorithms may be used to solve TSP. However, conventional approaches lead to exponential time or

unsatisfactory quality. To beat these shortcomings, many metaheuristic algorithms in the literature have been developed for TSP due to the importance of accomplishing improved solutions in realistic computing time [7].

Meta-heuristic algorithms can be classified into two major groups; single-based solution and population-based solution [6], [8], [9]. The ability of meta-heuristic algorithms to address optimization problems, such as TSP, relies on two elements: exploitation and exploration. Exploration refers to the research within the search space of unvisited regions, whereas exploitation refers to the search in the existing problem space regions for good solutions [10]–[12]. Single-based algorithms, including variable neighborhood search [13], simulated annealing [14], and guided local search [15], aim to improve a single candidate solution. By contrast, population-based algorithms maintain and enhance candidate solutions, often using population features to conduct direct search; such algorithms include biogeography-based optimization [16], grey wolf optimizer [17], particle swarm optimization (PSO) [18], emperor penguin colony [19], genetic algorithm (GA) [20], ant colony optimization (ACO) [21], black hole (BH) algorithm [22], and dragonfly algorithm (DA) [23]. In recent years, studies on plants have demonstrated that plants display intelligent behavior. Consequently, plants are believed to have a nervous system [24]. Examples of plant intelligence algorithms include the sapling growing up algorithm [25], rooted tree optimization [26], runner root algorithm [27], and strawberry algorithm as plant propagation algorithm (PPA) [28].

PPA was initially suggested by [28] to solve numerical problems; it emulates the survival technique adopted by plants, in which they survive by colonizing new areas with good growing conditions. The strawberry plant has a survival of sustainability and growth that send short runners to exploit the quest for good solutions in existing problem space regions and send long runners in the search space to explore unvisited regions. In [29], studied PPA to solve TSP and showed that PPA can produce better solutions than other algorithms. However, applying a deterministic local search based on 2-opt and k-opt takes exponential time to find an optimal solution; moreover, when this occurs slows down its convergence speed, simply because there may be a deficiency of diversity in certain solutions which leads not to thrust the algorithm towards optimal regions. Consequently, to ensure better convergence, and make the algorithm have solution diversity in both local and global search. This present study implements a crossover operation and three mutation operations (flip, swap, and slide) in PPA and the proposed termed as partial-partitioned greedy algorithm (PPGA). The main contribution of a scientific study is to improve PPA for solving TSP using TSPLIB and produce a promising variant of PPA. The proposed algorithm PPGA is evaluated using 10 TSP datasets (with different sizes and complexities) selected from traveling salesman problem library (TSPLIB). The proposed PPGA is also compared with five metaheuristic algorithms: ACO, PSO, GA, BH, and DA. The main advantage of PPGA is the ability to find an ideal or near-ideal solution in a short time.

This paper is structured being as: section 2 explains the mathematics of TSP. Section 3 provides the literature review. Afterward, section 4 discussed the proposed PPGA methods. Section 5 and 6 explores the experimental results, performance evaluation, and benchmark datasets used in this study are presented. Lastly, section 7 the conclusions and recommendations for potential future research are given.

2. TRAVELING SALESMAN PROBLEM

The importance of TSP is attributed to the detailed studies and high guidelines of computer scientists for it to be included in the assessment of modern optimization algorithms. This problem has been shown to be an NP-hard problem. It can be defined being as: An agent must visit N nodes exactly once and return at the starting node at the lowest expense, i.e., lowest time of visitation or the shortest distance. A cost matrix $C = [c_{ij}]$ is explored to obtain a permutation $\pi : \{0, \dots, N - 1\} \rightarrow \{0, \dots, N - 1\}$, where c_{ij} shows the expense of visiting node (j) from node (i). The aim is to reduce an objective function represented by $f(\pi, C)$ as shown in (1):

$$f(\pi, C) = \sum_{i=0}^{N-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + (c_{\pi(N)}, c_{\pi(1)}) \quad (1)$$

where $\pi(i)$ shows the i^{th} node in permutation π , d is the distance between nodes and $c_{ij}=c_{ji} \forall i, j$ and the position of city (i) can be verified by utilizing the values of the x-axes, and y-axes, i.e., x_i and y_i sequentially.

3. LITERATURE REVIEW

Various algorithms, including single and hybrid algorithms, for TSP have been developed. In [30] proposed a hybrid approach using PSO to improve ACO performance parameters. In addition, a 3-opt local

search was endowed to the proposed approach to enhance local search. However, the proposed algorithm has many operations that consume additional time in improving the same search regions to determine the best improvement. In [31] solved the issue of unstable nature instance problem by providing ACO with a local search operator. This operator iteratively chooses the best solution found by the algorithm and then continues to remove or insert cities to improve the solution quality. Nonetheless, using multi-operator in local search may radically increase the computation time or reduce their performance. GA has been implemented by several researchers for TSP. In [32] proposed the use of GA to resolve the challenging large-scale colored balanced TSP. nevertheless, the proposed next generation access (NGA) algorithm should demonstrate good performance in terms of solving speed or solution quality.

In accordance with [33] studied a hybrid metaheuristic algorithms address TSP based on simulated annealing (SA) and the symbiotic organisms search. The possible challenge of the proposed algorithm can be found to a few considerations, the proposed algorithm includes the use of several parameters. Additionally, increasing the problem complexity, the configuration of the algorithm is linked to it. In another related work [34] studied on improving the performance of the SA by using a greedy search to deal properly with large-scale TSP. However, the proposed algorithm stuck in local optima. This is because the SA utilizes a greedy acceptance criterion that only takes an optimized solution and excludes the worst solution. The probabilistic TSP was solved in multiple trials in [35] through an adaptive multiswarm PSO. In the suggested adaptive PSO, random values are allocated in the initial stage of the search. Next, these parameters are configured dynamically at the same time as the objective function of the problem is optimized. Nonetheless, the search process lacks feedback information and learning due to having less values of parameters to optimize. In [36] strengthened PSO to solve the imprecise cost matrix TSP. The PSO modifications consist of the adoption of the swap series, the swap process, and the guidelines for various speed updates. Nonetheless, several methods have been used in the proposed algorithm which affects to take additional time to get the best improvement. In [37] proposed a hybrid between firefly algorithm (FA) and GA; here, the distance of the FA is redefined by presenting two swap methods to prevent dropping into local optima. The created population which has poor solutions that could lead to long-term convergence to an ideal solution.

According to [38] investigated the BH algorithm to solve TSP. The implementation of the BH algorithm was assessed on 10 datasets and the outcomes in comparison with other optimization techniques. The computational results showed that the BH algorithm can provide solutions better than ACO, GA, and PSO. However, The BH algorithm still lacks the capability to perform high exploration during the update process. Due to a new solution is produced randomly when the previous solution is not improved. Furthermore, a similar study was conducted in [39] to investigate the DA on solving TSP. PPA has been investigated to work on discrete problems, specifically on TSP [29]. The research concerns the usage of the idea of long and short runners in maximum graphs while looking for Hamiltonian cycles. The performance of the PPA algorithm was tested on a traditional dataset and compared with that of PSO SA, GA, and FA. Experimental results were included; however, the performance of the algorithm in solving TSP must be further investigated and compared with that of other optimization methods. Besides, the PPA algorithm suffers from slows down its convergence speed, simply because there may be a deficiency of diversity in certain solutions which leads not to thrust the algorithm towards optimal regions.

4. PROPOSED PARTIAL-PARTITIONED GREEDY ALGORITHM FOR TRAVELING SALESMAN PROBLEM

PPGA randomly begins with the initial population of plants/tours/solutions and iteratively improvises solutions for a given problem instance. In each iteration, PPGA improvises solutions by using short and long runners. The PPGA algorithm proceeds as:

4.1. Initial population

The initial population is a collection of an ordered list of plants where every plant represents a sequence of cities. X_i is tour i , $i=1, \dots, NP$, implying that NP is the plant population size. In accordance with the Euclidean distance, tour lengths X_L are calculated. Each city of plant is assigned a label of city such that no city can be seen twice in the same plant. TSP tour representation primarily has two strategies: adjacency and path. In this study, path representation is chosen for a tour. As shown in Figure 1, let $\{A, B, C, D, E\}$ be the labels of cities where A is the starting point.

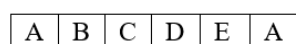


Figure 1. Plant/tour

4.2. Short runners (exploitation task)

A pre-determined tour set is taken from the ones that had short-length tours after sorting the tours by their tour lengths; from these tours, short runners are sent, i.e., new local tours are produced from them. Crossover is the synthesis of the two tours to create new tours that are copied into the new tours. The crossover operation, which utilizes the crossover simplest case, randomly chooses two tours to crossover, randomly picks a crossover point on the basis of (2), and then changes all cities after that point.

$$CP_{i...n} = r \cdot (L_{i...n} - 2) + 1 \quad (2)$$

where CP shows the crossover point, $r \in [0, 1]$ is a randomly selected index, and L is the size of plants.

4.3. Long runners (exploration task)

Long runners are implemented by using three mutation operations (flip, swap, and slide). To explain the flip, swap, and slide operations, A, B, C, D and E are considered cities in the tour visited in sequence $\{A \rightarrow B \rightarrow C \rightarrow D \rightarrow E\}$. The operation structures produced are presented as:

- Flip: Orders the cities vice versa from the last city to the first city; the new sequence is $\{A \rightarrow E \rightarrow D \rightarrow C \rightarrow B\}$.
- Swap: Swaps two cities in the tour, where B and E are exchanged the new sequence is $\{A \rightarrow E \rightarrow C \rightarrow D \rightarrow B\}$.
- Slide: Slides two cities (B and C) after all the other cities' positions; the new sequence is $\{A \rightarrow D \rightarrow E \rightarrow B \rightarrow C\}$.

PPGA started with a good population of tours (plants). The initial population diversity is supposed to be created by random methods utilized to create tours. Therefore, short runners conduct the exploitation process, and long runners control the exploration process in the search space. Figure 2 shows the pseudo-codes of the PPGA algorithm.

```

PPGA Algorithm
1. Create a population  $P = X_i$ ,  $i = 1, \dots, NP$  of valid tours; select values for  $g_{max}$  and  $y$ .
2.  $g = 1$ 
3. while  $g < g_{max}$  do
4.   Calculate  $N_i = f(X_i)$ ,  $\forall X_i \in P$ 
5.   Sort  $N = N_i$ ,  $i = 1, \dots, NP$  in ascending order (for minimization);
6.   for  $i = 1: E(NP/10)$ , Top 10 % of plants do
7.     Create  $(y/i)$  short runners and select two random plants  $i..i+1$  apply crossover
operator, where  $y$ 
is an arbitrary parameter.
8.     if  $N_{i..i+1} > f(r_{i..i+1})$  then
9.        $X_{i..i+1} \leftarrow r_{i..i+1}$ 
10.    else
11.     Ignore  $r_{i..i+1}$ 
12.    end if
13.  end for
14.  for  $i = E(NP/10) + 1: NP$  do
15.     $r_i = 1$  runner for plant  $i$  using random mutation operator, mutation operator = 3, 1 long
runner for each plant not in the top 10 percent
16.  if  $N_i > f(r_i)$  then
17.     $X_i \leftarrow r_i$ 
18.  else
19.    Ignore  $r_i$ 
20.  end if
21. end for
22. end while

```

Figure 2. Pseudo-codes of the PPGA algorithm

5. EXPERIMENTAL SETUP

This section presents the performance and robustness of PPGA in solving TSP. Two experimental results are used. The first experiment PPGA against discrete PPA. In the second experiment, the proposed PPGA is compared with five population-based algorithms, i.e., ACO, PSO, GA, BH, and DA. The proposed PPGA is tested on 10 benchmark TSP datasets with diverse characteristics taken from TSPLIB [3]. Choosing different instance structures provides great insights into the behavior of the proposed algorithm when

addressing TSP. Figure 3 simplifies one type of TSP instances extracted from the TSPLIB benchmark library.

```

NAME: ulysses22.tsp
TYPE: TSP
COMMENT: Odyssey of Ulysses (Groetschel/Padberg)
DIMENSION: 22
EDGE_WEIGHT_TYPE: GEO
DISPLAY_DATA_TYPE: COORD_DISPLAY
NODE_COORD_SECTION
1 38.24 20.42
2 39.57 26.15
3 40.56 25.32
4 36.26 23.12
5 33.48 10.54
6 37.56 12.19
7 38.42 13.11
8 37.52 20.44
9 41.23 9.10
10 41.17 13.05
11 36.08 -5.21
12 38.47 15.13
13 38.15 15.35
14 37.51 15.17
15 35.49 14.32
16 39.36 19.56
17 38.09 24.36
18 36.09 23.00
19 40.44 13.57
20 40.33 14.15
21 40.37 14.23
22 37.57 22.56
EOF

```

Figure 3. Sample structure of the Ulysses22 dataset

In all datasets, n nodes reflect unique positions in specific towns, e.g., Berlin. The first five lines provide some details, such as the data type (including Euclidean, geographical or other forms of data) about the problem being discussed. The keyword TYPE defines the category of data, e.g., symmetric, asymmetric, or tour set. The keyword DIMENSION is the number of nodes for TSP datasets. The keyword EDGE WEIGHT TYPE determines how the edge weight is described, e.g., the keyword EUC 2D is the Euclidean distance in the plane, whereas GEO is the geographical distance. The node coordinate part starts with the keyword NODE COORD_SECTION. The node identifier, x and y coordinates are made up of every line. The identifier of the node is a unique integer ≥ 1 . Table 1 summarizes the statistics for several TSP instances.

Table 1. Description of some TSP instances

| Data name | Location |
|-----------|------------------------------|
| ulysses22 | Groetschel/Padberg |
| bays29 | Groetschel, Juenger, Reinelt |
| bayg29 | Groetschel, Juenger, Reinelt |
| att48 | Padberg/Rinaldi |
| eil51 | Christofides/Eilon |
| berlin52 | Berlin (Germany) |
| st70 | Smith/Thompson |
| eil76 | Christofides/Eilon |
| gr96 | Europe |
| eil101 | Christofides/Eilon |

The proposed PPGA algorithm is implemented in the programming environment MATLAB 2020a and executed in an Intel® Core™ i5 CPU, 2.40 GHz, 4 GB RAM memory, and Windows 7. Each test is conducted of five independent runs. The maximum number of generations (g_{max}) and population size are set to 200 and 100, respectively, for the proposed PPGA algorithm.

6. EXPERIMENTAL RESULTS

The experimental results of the proposed PPGA is compared with PPA, as shown in Table 2. The best, worst, average, standard deviation (Std), and time (in seconds) are listed for each algorithm. Table 2 compares the results of the proposed PPGA against PPA for ten various datasets TSP. All the problems have been run five times independently.

Table 2. Comparison of PPGA against PPA

| Dataset | Algorithm | Best | Worst | Average | Std | Time |
|-----------|-----------|----------|----------|----------|---------|-------|
| ulysses22 | PPA | 66.81 | 84.16 | 73.33 | 6.92 | 1.05 |
| | PPGA | 75.30 | 76.58 | 75.92 | 0.56 | 4.86 |
| bays29 | PPA | 9883.32 | 10504.77 | 10257.19 | 265.89 | 1.06 |
| | PPGA | 9105.87 | 9764.46 | 9311.17 | 287.96 | 6.44 |
| bayg29 | PPA | 9253.00 | 10823.40 | 10070.52 | 763.67 | 1.06 |
| | PPGA | 9120.33 | 9568.70 | 9318.19 | 178.09 | 2.57 |
| att48 | PPA | 42308.39 | 47767.24 | 44816.51 | 2020.77 | 1.09 |
| | PPGA | 33961.11 | 34581.00 | 34585.88 | 495.81 | 9.33 |
| eil51 | PPA | 504.75 | 604.33 | 535.98 | 39.11 | 1.10 |
| | PPGA | 444.03 | 458.42 | 450.39 | 5.20 | 9.91 |
| berlin52 | PPA | 8971.38 | 10420.33 | 9864.75 | 537.23 | 1.11 |
| | PPGA | 7748.63 | 8359.85 | 8190.14 | 261.48 | 10.00 |
| st70 | PPA | 924.72 | 1125.70 | 1045.35 | 74.88 | 1.10 |
| | PPGA | 714.36 | 804.27 | 770.25 | 34.71 | 13.14 |
| eil76 | PPA | 740.36 | 841.25 | 786.78 | 44.17 | 1.16 |
| | PPGA | 586.47 | 632.15 | 604.68 | 17.33 | 14.35 |
| gr96 | PPA | 1014.70 | 1067.18 | 1038.06 | 21.55 | 1.17 |
| | PPGA | 634.36 | 721.80 | 670.88 | 34.65 | 17.92 |
| eil101 | PPA | 1066.95 | 1194.35 | 1137.02 | 60.34 | 1.20 |
| | PPGA | 774.68 | 835.36 | 796.48 | 27.11 | 18.85 |
| Average | PPA | 7473.43 | 8443.27 | 7962.54 | 383.45 | 1.11 |
| | PPGA | 6316.51 | 6580.25 | 6477.39 | 134.29 | 10.73 |

According to the results in Table 2 column 3, the proposed PPGA algorithm achieved nine out of ten datasets, which means 90% better than PPA in terms of best distance. The average comparison was shown in the lower section of the table. Table 2 results indicate that PPGA was better than PPA, according to the average tour costs and the average standard deviation for all datasets shown in column five and six. For the ten selected datasets, the average cost of touring PPGA was 6477.39. The achieved average tour costs for PPA was 7962.54. According to Table 2, this result is due to the improvement process achieved by the crossover and three mutation operations (flip, swap, and slide) which overcome the problem of slows down its convergence speed, and deficiency of diversity in discrete PPA.

Owing to the fact that there are a vast number of articles suggested for TSP in the literature, this study selected the algorithms that were recently published and those that obtained the best outcomes using the same datasets in the experiment to be compared with them in this study. The proposed PPGA is compared with five algorithms that are available in the literature: i.g., ACO, PSO, GA, and BH by [38] and DA proposed by [39]. The computational results are presented in Table 3. The best, worst, average, standard deviation (Std), and time (in seconds) are stated for each algorithm in Table 3.

Table 3 illustrates data showing that PPGA obtains the best results in six datasets out of 10 datasets, which means 60% better than ACO. PPGA outperforms PSO, GA, and DA in all 10 datasets. PPGA also achieves better outcomes in 5 datasets and equal results in one dataset than BH. Small values demonstrate the best solutions obtained and vice versa. In addition, for each algorithm, the Std on various runs is given to demonstrate algorithm efficiency and stability. A description of the average comparison is shown in the lower part of the table. Along with the results given in Table 3, the average tour costs in column five for all the datasets prove that PPGA is better than ACO, PSO, GA, BH, and DA. Based on the results in Table 3, the proposed PPGA algorithm superiority other algorithms in some data and rival in other data this is due to the improvement process achieved in PPGA by the crossover and three mutation operations (flip, swap, and slide). Where crossover and three mutation operations (flip, swap, and slide) have a responsibility in the balance between exploitation and exploration. The average tour cost for PPGA is 6477.39 for the selected ten problems. The accomplished average tour costs for ACO, PSO, GA, BH, and DA are 7089.73, 8307.81, 7661.78, 6546.42, and 6994.93, respectively. Moreover, the standard solution deviation achieved by the PPGA algorithm is slightly worse than DA, but better than ACO, PSO, GA, and BH. This result implies that in seeking optimal solutions, the PPGA algorithm is more efficient and robust, whereas other algorithms such

as ACO, PSO, GA, and BH may be stuck in local optimal solutions. Finally, the proposed PPGA algorithm achieved a better average execution time than other optimization algorithms.

Table 3. Comparison of PPGA versus other algorithms

| Dataset | Algorithm | Best | Worst | Average | Std | Time |
|-----------|-----------|----------|----------|----------|---------|--------|
| ulysses22 | ACO | 75.39 | 75.84 | 75.48 | 0.19 | 84.27 |
| | PSO | 75.91 | 77.18 | 76.21 | 0.55 | 61.87 |
| | GA | 75.77 | 76.44 | 75.98 | 1.23 | 63.39 |
| | BH | 75.30 | 75.93 | 75.68 | 0.34 | 50.44 |
| | DA | 76.82 | 80.93 | 77.83 | 1.16 | 21.00 |
| | PPGA | 75.30 | 76.58 | 75.92 | 0.56 | 4.86 |
| bays29 | ACO | 9239.19 | 11014.44 | 9823.20 | 722.41 | 88.25 |
| | PSO | 9120.33 | 9498.17 | 9195.90 | 168.97 | 88.82 |
| | GA | 9751.42 | 10513.91 | 10015.23 | 319.87 | 57.11 |
| | BH | 9396.47 | 9507.17 | 9463.25 | 60.95 | 52.10 |
| | DA | 9387.03 | 9611.78 | 9480.29 | 64.96 | 22.00 |
| | PPGA | 9105.87 | 9764.46 | 9311.17 | 287.96 | 6.44 |
| bayg29 | ACO | 9447.49 | 11033.54 | 9882.21 | 675.83 | 99.95 |
| | PSO | 9329.25 | 11332.72 | 9947.02 | 799.40 | 75.29 |
| | GA | 9579.12 | 10411.19 | 9771.95 | 127.11 | 56.16 |
| | BH | 9375.44 | 9375.44 | 9375.44 | 0.00 | 45.87 |
| | DA | 9464.41 | 9704.98 | 9547.75 | 64.75 | 22.00 |
| | PPGA | 9120.33 | 9568.70 | 9318.19 | 178.09 | 2.57 |
| att48 | ACO | 35230.90 | 46204.24 | 39436.18 | 4874.29 | 133.45 |
| | PSO | 36996.44 | 61421.99 | 47018.41 | 9685.89 | 84.73 |
| | GA | 35312.51 | 50671.45 | 43620.63 | 2004.00 | 57.35 |
| | BH | 34200.86 | 35528.51 | 34473.84 | 589.80 | 43.21 |
| | DA | 37225.85 | 38683.21 | 37759.73 | 425.69 | 23.00 |
| | PPGA | 33961.11 | 34581.00 | 34585.88 | 495.81 | 9.33 |
| eil51 | ACO | 454.38 | 469.05 | 461.01 | 6.29 | 59.19 |
| | PSO | 469.15 | 737.52 | 574.80 | 107.23 | 57.25 |
| | GA | 448.83 | 462.11 | 453.47 | 9.41 | 59.63 |
| | BH | 437.89 | 526.89 | 458.92 | 38.63 | 44.39 |
| | DA | 471.56 | 491.65 | 475.16 | 4.51 | 23.00 |
| | PPGA | 444.03 | 458.42 | 450.39 | 5.20 | 9.91 |
| berlin52 | ACO | 7757.02 | 10541.12 | 8522.90 | 1152.2 | 65.07 |
| | PSO | 9218.46 | 14279.43 | 11089.52 | 2067.93 | 68.64 |
| | GA | 8779.75 | 9565.37 | 9288.44 | 1301.21 | 52.73 |
| | BH | 8188.07 | 9356.74 | 8455.83 | 508.98 | 43.40 |
| | DA | 9400.75 | 9610.15 | 9486.70 | 72.54 | 23.00 |
| | PPGA | 7748.63 | 8359.85 | 8190.14 | 261.48 | 10.00 |
| st70 | ACO | 711.65 | 855.20 | 757.75 | 59.60 | 94.56 |
| | PSO | 1030.84 | 1756.12 | 1321.81 | 269.27 | 55.28 |
| | GA | 1112.30 | 1242.20 | 1158.84 | 52.17 | 55.09 |
| | BH | 723.26 | 1081.10 | 797.57 | 125.22 | 45.33 |
| | DA | 797.47 | 887.08 | 839.01 | 24.28 | 29.00 |
| | PPGA | 714.36 | 804.27 | 770.25 | 34.71 | 13.14 |
| eil76 | ACO | 574.24 | 665.99 | 594.14 | 40.21 | 61.74 |
| | PSO | 804.26 | 1195.90 | 975.63 | 152.40 | 56.76 |
| | GA | 619.22 | 679.78 | 652.05 | 122.09 | 46.69 |
| | BH | 566.24 | 925.84 | 659.10 | 152.17 | 46.54 |
| | DA | 624.92 | 674.48 | 644.89 | 13.03 | 30.00 |
| | PPGA | 586.47 | 632.15 | 604.68 | 17.33 | 14.35 |
| gr96 | ACO | 555.75 | 639.91 | 580.54 | 33.93 | 84.38 |
| | PSO | 1095.11 | 1728.82 | 1378.86 | 247.50 | 56.21 |
| | GA | 737.96 | 748.35 | 742.42 | 4.32 | 63.24 |
| | BH | 546.83 | 1197.87 | 807.24 | 258.81 | 43.58 |
| | DA | 671.00 | 836.00 | 734.05 | 47.26 | 40.00 |
| | PPGA | 634.36 | 721.80 | 670.88 | 34.65 | 17.92 |
| eil101 | ACO | 725.09 | 868.20 | 763.92 | 59.96 | 89.63 |
| | PSO | 1158.70 | 1973.81 | 1499.99 | 319.74 | 62.09 |
| | GA | 828.88 | 854.43 | 838.83 | 9.96 | 55.18 |
| | BH | 720.38 | 1249.86 | 897.38 | 210.14 | 45.83 |
| | DA | 812.80 | 997.60 | 898.52 | 47.90 | 36.00 |
| | PPGA | 774.68 | 835.36 | 796.48 | 27.11 | 18.85 |
| Average | ACO | 6477.11 | 8236.75 | 7089.73 | 762.49 | 86.04 |
| | PSO | 6929.84 | 10400.17 | 8307.81 | 1381.88 | 66.69 |
| | GA | 6724.57 | 8522.52 | 7661.78 | 395.13 | 56.65 |
| | BH | 6423.07 | 6882.53 | 6546.42 | 194.50 | 46.06 |
| | DA | 6893.26 | 7157.78 | 6994.93 | 76.60 | 26.9 |
| | PPGA | 6316.51 | 6580.25 | 6477.39 | 134.29 | 10.73 |

7. CONCLUSION

For several decades, the creation of intelligent behavior in plants to solve complex problems has been an important research area. In this study, discrete PPA is improved to solve COPs, particularly the TSP. The algorithm is evaluated on ten benchmark datasets and compared with five common algorithms in the literature to determine the efficacy of the proposed PPGA algorithm for solving TSP. The experimental results indicate that, relative to ACO, PSO, GA, BH, and DA, the PPGA algorithm can yield high-quality solutions. Moreover, the experimental results demonstrate that PPGA considerably outperforms other algorithms in terms of average tour cost and execution time. Further research concerns extending PPGA to solve other COPs such as, the facility layout problem, the quadratic assignment problem, and vehicle routing problems.





REFERENCES

- [1] T. Sahai, "Dynamical systems theory and algorithms for NP-hard problems," *Studies in Systems, Decision and Control*, vol. 304, pp. 183–206, 2020, doi: 10.1007/978-3-030-51264-4_8.
- [2] S. Mortada and Y. Yusof, "A neighbourhood search for artificial bee colony in vehicle routing problem with time windows," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 3, pp. 255–266, Jun. 2021, doi: 10.22266/ijies2021.0630.22.
- [3] G. Reinelt, "TSPLIB. A traveling salesman problem library," *ORSA journal on computing*, vol. 3, no. 4, pp. 376–384, 1991, doi: 10.1287/ijoc.3.4.376.
- [4] R. Sagban, K. R. Ku-Mahamud, and M. S. Abu Bakar, "Reactive max-min ant system with recursive local search and its application to TSP and QAP," *Intelligent Automation and Soft Computing*, vol. 23, no. 1, pp. 127–134, 2017, doi: 10.1080/10798587.2016.1177914.
- [5] A. Kumar, "Improved genetic algorithm to solve small scale travelling salesman problem," in *Proceedings of the International Conference on Intelligent Computing and Control Systems, ICIACS 2020*, 2020, pp. 516–520, doi: 10.1109/ICIACS48265.2020.9120880.
- [6] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, "Metaheuristic research: a comprehensive survey," *Artificial Intelligence Review*, vol. 52, no. 4, pp. 2191–2233, 2019, doi: 10.1007/s10462-017-9605-z.
- [7] D. Karaboga and B. Gorkemli, "Solving traveling salesman problem by using combinatorial artificial bee colony algorithms," *International Journal on Artificial Intelligence Tools*, vol. 28, no. 1, 2019, doi: 10.1142/S0218213019500040.
- [8] H. N. K. Al-Behadili, K. R. Ku-Mahamud, and R. Sagban, "Genetic-based pruning technique for ant-miner classification algorithm," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 11, no. 1, p. 304, Feb. 2021, doi: 10.18517/ijaseit.11.1.10826.
- [9] H. N. K. AL-Behadili, K. R. Ku-Mahamud, and R. Sagban, "Hybrid ant colony optimization and genetic algorithm for rule induction," *Journal of Computer Science*, vol. 16, no. 7, pp. 1019–1028, 2020, doi: 10.3844/JCSP.2020.1019.1028.
- [10] A. M. Jabbar, K. R. Ku-Mahamud, and R. Sagban, "An improved ACS algorithm for data clustering," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 17, no. 3, pp. 1506–1515, 2020, doi: 10.11591/ijeecs.v17.i3.pp1506-1515.
- [11] A. Ullah, N. M. Nawli, J. Uddin, S. Baseer, and A. H. Rashed, "Artificial bee colony algorithm used for load balancing in cloud computing: Review," *IAES International Journal of Artificial Intelligence*, vol. 8, no. 2, pp. 156–167, Jun. 2019, doi: 10.11591/ijai.v8.i2.pp156-167.
- [12] A. M. Jabbar, K. R. Ku-Mahamud, and R. Sagban, "Improved self-adaptive ACS Algorithm to determine the optimal number of clusters," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 11, no. 3, pp. 1092–1099, 2021, doi: 10.18517/ijaseit.11.3.11723.
- [13] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers and Operations Research*, vol. 24, no. 11, pp. 1097–1100, Nov. 1997, doi: 10.1016/S0305-0548(97)00031-2.
- [14] P. J. M. van Laarhoven and E. H. L. Aarts, "Simulated annealing," in *Simulated Annealing: Theory and Applications*, Dordrecht: Springer Netherlands, 1987, pp. 7–15.
- [15] C. Voudouris and E. P. K. Tsang, "Guided Local Search," in *Handbook of Metaheuristics*, vol. 1–2, Boston: Kluwer Academic Publishers, 2018, pp. 185–218.
- [16] D. Simon, "Biogeography-based optimization," *IEEE transactions on evolutionary computation*, vol. 12, no. 6, pp. 702–713, 2008.
- [17] H. Almazini and K. Ku-Mahamud, "Grey wolf optimization parameter control for feature selection in anomaly detection," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 2, pp. 474–483, Apr. 2021, doi: 10.22266/ijies2021.0430.43.
- [18] T.-A. N. Abdali, R. Hassan, R. C. Muniyandi, A. H. Mohd Aman, Q. N. Nguyen, and A. S. Al-Khaleefa, "Optimized particle swarm optimization algorithm for the realization of an enhanced energy-aware location-aided routing protocol in MANET," *Information*, vol. 11, no. 11, p. 529, Nov. 2020, doi: 10.3390/info11110529.
- [19] S. Harifi, M. Khalilian, J. Mohammadzadeh, and S. Ebrahimnejad, "Emperor penguins colony: a new metaheuristic algorithm for optimization," *Evolutionary Intelligence*, vol. 12, no. 2, pp. 211–226, 2019, doi: 10.1007/s12065-019-00212-x.
- [20] M. Mitchell, "An introduction to genetic algorithms," *An Introduction to Genetic Algorithms*, vol. 1996, 2020, doi: 10.7551/mitpress/3927.001.0001.
- [21] H. Almazini and K. Ku-Mahamud, "Adaptive technique for feature selection in modified graph clustering-based ant colony optimization," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 3, pp. 332–345, Jun. 2021, doi: 10.22266/ijies2021.0630.28.
- [22] A. Hatamlou, "Black hole: a new heuristic optimization approach for data clustering," *Information Sciences*, vol. 222, pp. 175–184, 2013, doi: 10.1016/j.ins.2012.08.023.
- [23] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing and Applications*, vol. 27, no. 4, pp. 1053–1073, 2016, doi: 10.1007/s00521-015-1920-1.
- [24] S. Akyol and B. Alatas, "Plant intelligence based metaheuristic optimization algorithms," *Artificial Intelligence Review*, vol. 47, no. 4, pp. 417–462, 2017, doi: 10.1007/s10462-016-9486-6.





- [25] A. Karci, "Theory of saplings growing up algorithm," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2007, vol. 4431 LNCS, no. PART 1, pp. 450–460, doi: 10.1007/978-3-540-71618-1_50.
- [26] Y. Labbi, D. Ben Attous, H. A. Gabbar, B. Mahdad, and A. Zidan, "A new rooted tree optimization algorithm for economic dispatch with valve-point effect," *International Journal of Electrical Power and Energy Systems*, vol. 79, pp. 298–311, 2016, doi: 10.1016/j.ijepes.2016.01.028.
- [27] F. Merrikh-Bayat, "The runner-root algorithm: a metaheuristic for solving unimodal and multimodal optimization problems inspired by runners and roots of plants in nature," *Applied Soft Computing Journal*, vol. 33, pp. 292–303, 2015, doi: 10.1016/j.asoc.2015.04.048.
- [28] A. Salhi and E. S. Fraga, "Nature-inspired optimisation approaches and the new plant propagation algorithm," in *International Conference on Numerical Analysis and Optimization (ICeMATH)*, 2011.
- [29] B. Selamoğlu and A. Salhi, "The plant propagation algorithm for discrete optimisation: The case of the travelling salesman problem," in *Studies in Computational Intelligence*, vol. 637, Springer, 2016, pp. 43–61.
- [30] M. Mahi, Ö. K. Baykan, and H. Kodaz, "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem," *Applied Soft Computing Journal*, vol. 30, pp. 484–490, 2015, doi: 10.1016/j.asoc.2015.01.068.
- [31] M. Mavrovouniotis, F. M. Muller, and S. Yang, "Ant colony optimization with local search for dynamic traveling salesman problems," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1743–1756, 2017, doi: 10.1109/TCYB.2016.2556742.
- [32] X. Dong and Y. Cai, "A novel genetic algorithm for large scale colored balanced traveling salesman problem," *Future Generation Computer Systems*, vol. 95, pp. 727–742, 2019, doi: 10.1016/j.future.2018.12.065.
- [33] A. E. S. Ezugwu, A. O. Adewumi, and M. E. Frincu, "Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem," *Expert Systems with Applications*, vol. 77, pp. 189–210, 2017, doi: 10.1016/j.eswa.2017.01.053.
- [34] X. Wu and D. Gao, "A study on greedy search to improve simulated annealing for large-scale traveling salesman problem," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2017, vol. 10386 LNCS, pp. 250–257, doi: 10.1007/978-3-319-61833-3_26.
- [35] Y. Marinakis, M. Marinaki, and A. Migdalas, "Adaptive tuning of all parameters in a multi-swarm particle swarm optimization algorithm: An application to the probabilistic traveling salesman problem," *Springer Proceedings in Mathematics and Statistics*, vol. 130, pp. 187–207, 2015, doi: 10.1007/978-3-319-18567-5_10.
- [36] I. Khan, S. Pal, and M. K. Maiti, "A modified particle swarm optimization algorithm for solving traveling salesman problem with imprecise cost matrix," in *Proceedings of the 4th IEEE International Conference on Recent Advances in Information Technology, RAIT 2018*, 2018, pp. 1–8, doi: 10.1109/RAIT.2018.8389060.
- [37] L. Teng and H. Li, "Modified discrete firefly algorithm combining genetic algorithm for traveling salesman problem," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 16, no. 1, pp. 424–431, 2018, doi: 10.12928/TELKOMNIKA.V16I1.4752.
- [38] A. Hatamlou, "Solving travelling salesman problem using black hole algorithm," *Soft Computing*, vol. 22, no. 24, pp. 8167–8175, 2018, doi: 10.1007/s00500-017-2760-y.
- [39] A. I. Hammouri, E. T. A. Samra, M. A. Al-Betar, R. M. Khalil, Z. Alasmer, and M. Kanan, "A dragonfly algorithm for solving traveling salesman problem," in *Proceedings - 8th IEEE International Conference on Control System, Computing and Engineering, ICCSCE 2018*, 2019, pp. 136–141, doi: 10.1109/ICCSCE.2018.8684963.

BIOGRAPHIES OF AUTHORS






Hussein Fouad Almazini     is a Ph.D student at School of Computing, Sintok, Universiti Utara Malaysia, Kedah, Malaysia. He holds a master's degree in information technology in the area (Artificial Intelligence). He works in a artificial intelligence, evolutionary computation, meta-heuristic algorithms and swarm intelligence. He is also interested in mining scientific datasets in domains such as IoT, healthcare, cyber security, social network, business, demographic, sustainability, and intelligence analysis. He can be contacted at email: h.almazni22@gmail.com.






Salah Mortada     is a Ph.D student at School of Computing, Sintok, Universiti Utara Malaysia, Kedah, Malaysia. He holds a master's degree in information technology in the area (Usability and User Experience). His current research interests include artificial intelligence, evolutionary computation, meta-heuristic algorithms, swarm intelligence, and their applications in the design and optimization of intelligent transportation management such as VRPs. He can be contacted at email: sms099@yahoo.com.






Hassan Fouad Almazini    is a Ph.D. student at School of Computing, Sintok, Universiti Utara Malaysia, Kedah, Malaysia. He holds a master's degree in information technology in the area (Artificial Intelligence). Hassan research and development experience includes over 8 years in the academia and Industry. He works in a artificial intelligence, evolutionary computation, meta-heuristic algorithms and swarm intelligence. He is also interested in mining scientific datasets in domains such as IoT, healthcare, cyber security, social network, business, demographic, sustainability, and intelligence analysis. He can be contacted at email: abba@ahsgs.uum.edu.my.



Dr. Hayder Naser Khraibet Al-Behadili    has a Ph.D. degree in Information Technology specialization in artificial intelligence. His research is mainly focused on development of artificial intelligence and data mining techniques based on swarm intelligence algorithms. He can be contacted at email: hayderkhraibet@sa-uc.edu.iq.



Jawad Kadum Hussein    is a master student in Information System specialization in data mining. His research is mainly focused on improvement the data mining based on artificial intelligence, evolutionary computation, meta-heuristic algorithms, and swarm intelligence. He can be contacted at email: Jawadalkenani@sa-us.edu.iq.