https://doi.org/10.26682/sjuod.2023.26.1.29
Journal of University of Duhok., Vol. 26, No.1 (Pure and Engineering Sciences),Pp 291-304, 2023

A LEARNER PERFORMANCE-BASED BEHAVIOR ALGORITHM FOR SOLVING TRAVELLING SALESMAN PROBLEM

SHAHAD A. SALIH [*] and TARIK A. RASHID [**]
[*] Dept. of Computer Science, University of Duhok, Kurdistan Region–Iraq
[**] Dept. of Computer Science and Engineering, University of Kurdistan Hewler, Erbil, Kurdistan Region–Iraq

(Received: January 21, 2023; Accepted for Publication: April 2, 2023)

ABSTRACT

The traveling salesman problem (TSP), is one of the complex optimization problems. TSP is considered as a non-deterministic polynomial-time hardness (NP) hard problem, since its solution cannot be found in certain polynomial time. In this work, a new metaheuristic optimization algorithm called learner performance-based behavior (LPB) was applied to find a feasible solution for TSP. The LPB algorithm's operators are based on those of genetic algorithms (crossover and mutation). In addition, the genetic algorithm and particle swarm optimization algorithm were applied to TSP to compare the efficiency of the LPB algorithm with them to solve this problem. The LPB outperformed the generic algorithm and the particle swarm algorithm. For instance, when the number of nodes is 30, the population size is 100, and the number of iterations is 500 within a loop of 30 iterations, the minimum averages were: the LPB (655.1900067), the GA (723.084896), and the PSO (692.6886233).

KEY WORDS: Traveling salesman problem, Learner performance-based behavior, Optimization algorithms, Metaheuristics, Genetic algorithm, Particle swarm algorithm.

1. INTRODUCTION

Metaheuristics is a strategy used to find the approximate optimal solution to optimization problems using optimization algorithms (Gunantara and Putra, 2019). Genetic algorithm (GA) and particle swarm optimization (PSO) are the most popular types of meta-heuristic algorithms.

GA is a subset of a much larger computing field known as evolutionary computation. John Holland and his co-workers created GA in the early 1960s. It is a search strategy based on Charles Darwin's theory of evolution and biological system selection. Moreover, it is a population-based algorithm that used to solve optimization issues by mimicking organisms' evolutionary behavior. This population is progress from a random population of solutions by selection, crossover, and mutation operators inspired by natural evolution. By carrying out the stated set of actions, the population goes through an iterative procedure in which it achieves various stages, each of which is referred to as a generation. As a result of this method, it is believed that the population would reach a generation with a reasonable solution to the problem. The solution to the issue is written as a string of bits or real numbers in GA.

PSO is a swarm-based optimization approach presented by Eberhart and Kennedy (1995). It replicates the social behaviors of animals such as insects, plants, birds, and fishes. These swarms conform a cooperative manner to find food, and each member of the swarms changes the search pattern based on its own and other members' learning experiences. The PSO algorithm's main design concept is closely tied to two studies: PSO, like evolutionary algorithms, employs a swarm mode, which allows it to concurrently explore a vast region in the solution space of the optimized objective function.

LPB is an evolutionary optimization algorithm proposed by Chnoor and Tarik (2021). It is a population-based algorithm that was inspired by the mechanism of selecting students in colleges according to their grade point average. In the algorithm exploitation phase, a group of individuals is selected from the population. These individuals are then divided into subgroups, and the best individuals are then selected from the subgroups depending on their

fitness.This algorithm was based on GA operations (crossover and mutation) in its exploration phase. The two operations will be performed on the selected subgroup and the population to obtain a new generation of improved results (Rahman., & Rashid, 2021).

The traveling salesman problem is considered one of the most complex computational problems. it has been used in the fields of algorithm testing and optimization. It is based on finding the shortest and most effective route to visit a list of specific destinations (cities) and then return to the starting point. Each destination (city) will be visited only once (Reinelt, 2003). TSP has a wide search area to find the nearest optimal solution, and those are difficult to solve (Reinelt, 2003) (Fogel, 1988) (Larrañaga and Murga, et al. 1999). It is one of the most important ways to compare and measure the efficiency of algorithms, due to its simple formulation and high particle importance (Cheng and Mao, 2007). Also, it has been employed in several fields and applications such as logistics service, drilling holes in circuit boards, vehicle routing, and packet routing in the global system for mobile communications.

This paper solved TSP optimization problem using a new optimization algorithm namely learner performance-based behavior (LPB). That approved better results compared with GA and PSO algorithms, were applied on 30 nodes (city) and 500 iterations within loop 30 iterations.

The paper is organized as follows: Section 2 presents the related works that have used the optimization algorithms to solve TSP, whereas Section 3 provides travelling salesman problem mathematical representation. The methodology of LPB algorithm and applying it on TSP is presented in Section 4. The simulation experiments are shown in Section 5. Sections 6 and 7 discuss and conclude this paper.

2. RELATED WORKS

Several optimization algorithms were used to solve the traveling salesman problem. such as GA and PSO.

A novel method for GA population seeding, designed specifically for solving TSP, has been proposed by (Hassanat, A.B. and el at. 2018). A regression line and its perpendicular line are used to break the TSP problem into smaller sub-problems. The authors' clustered the cities into four sub-problems repeatedly based on their locations, until the sub-problems became small enough (typically containing four or fewer cities). These cities are most often neighbors, therefore joining them can result in a reasonable starting solution that is then repeatedly altered to build the initial population. The research offers enhancing the selection operator and applying the elite retention approach to improve the quality of selection in genetic algorithms. Additionally, using an adaptive algorithm selection in mutation operations can improve search results and variation. Additionally, the authors suggest adding a reverse operation following one-way chromosomal evolution to enable kids to acquire parental genes that enhance quality, enhancing the algorithm's capacity to find the best answer (Fu, C., and Qiao, 2018). In the same context, a new genetic algorithm has been developed by (Kaabi, J. and Harrath, 2019) to address the symmetric traveling salesman problem. The developed algorithm merged genetic algorithms with several heuristics to tackle the problem. Multiple crossover operators and mutation operators were tried out, and dynamic rates were suggested for both to make the proposed method work better. The average and maximum fitness of the previous population served as the basis for calculating these rates for each new group of people. Similarly, (Hariyadi, Nguyen and el at. 2020) employed the GA to determine the optimal path for a given set of cities, even when the number of cities is large. The GA can find the optimal global value for the total distance traveled over multiple iterations. The authors stated that the starting point for the journey can be any city. If the lowest fitness value is reached more than once over a number of generations, the algorithm will stop.

The user decides which generation is best, an d the system will stop if it does so. Fitness is determined by random values generated during natural selection. (Khan, Pal and el at. 2018) modified a version of PSO that can be used to solve the TSP with a cost matrix that is either crisp or fuzzy. The modified PSO technique uses swap sequences, swap operations, and various velocity update rules. Experiments indicate that the effectiveness of the aforementioned approach for solving such problems (Symmetric TSP and Asymmetric TSP) in terms of accuracy and execution time is superior to that of several other well-established algorithms for TSPs. In another study, an improvement was introduced in (Yu, M., 2019) to the ant colony algorithm by incorporating the PSO algorithm. The improved

ant colony algorithm was achieved by assigning a specific "particle property" to the ant colony, which could be used to solve the TSP. The proposed approach aims to optimize the architecture of the PSO algorithm for the TSP. It integrates multiple efficient components, each with its own contributions towards enhancing the effectiveness of the algorithm. Alternatively, the study in (Emambocus, and Amphawan, 2021) offers an enhanced swap sequence based PSO algorithm to increase the performance of the swap sequence based PSO by incorporating the strategies of the Expanded PSO into the swap sequence based PSO. In order to assess the suggested method, the solutions to the TSP problem generated from the proposed algorithm and swap sequence based PSO are compared in terms of the best solution, the mean solution, and the time required to converge to the optimum solution. In comparison to the swap sequence-based PSO, the suggested method provides superior solutions with shorter pathways when applied to TSP. Nevertheless, when applied to TSP, the swap sequence-based PSO is observed to converge quicker than the suggested approach. The authors in (Wei, B., and Gui, 2021) introduced the probability initialization strategy and genetic operator into the particle swarm optimization algorithm. They mentioned that several issues in particle swarm optimization, such as premature convergence and easy descent into suboptimal solutions, may be resolved. For tackling the traveling salesman issue, they suggested an enhanced hybrid particle swarm optimization to deal with TSP. The suggested method demonstrates superior performance on big scale TSP compared to small scale TSP.

In general, GA and PSO had some limitations. In one hand, the GA has a difficulty to model an objective fitness function in complex circumstances. Poor fitness functions can either converge on local optimum values or just disintegrate and diverge into nothingness, restricting generational advancement. For complex situations, dysfunctional and more complicated problems necessitate more computational power and more complex fitness models. These can be challenging to create, and sophisticated models can be exponentially more complicated to compute. Uncertain stop criteria make it difficult to establish when a complicated problem has reached a global optimum. When a genetic algorithm progresses toward greater fitness in later generations, it frequently hits a plateau. When a fitness plateau is mistaken for a global optimum, bad solutions are returned. On the other hand, it is easy for the PSO to get stuck in a local optimum in a high-dimensional space, and it has a low convergence rate during the iterative process.

3. TRAVELLING SALESMAN PROBLEM

Due to its difficult solving, but conceptually straightforward nature, the TSP is probably the combinatorial optimization problem that receives the most attention. It is an entire NP problem. A classical TSP is a problem in which each node must be visited once from a starting point so that the total distance travelled is minimized. The following mathematical is the explanation of TSP:

let us consider a graph (Matai, and Mittal, M.L., 2010):

G = (V, E) Where V is a set of cities and E is a set of weighted edges (Cost)

Minimize:

$$Cij(i, j) \in E$$

Subject:

$$\sum_{n\neq 0} CijXij \tag{3.1}$$

$$\sum_{j=1}^{n} Xij = 1 \quad (i \in V, i \neq j) \tag{3.2}$$

$$\sum_{i=1}^{n} Xij = 1 \quad (ji \in V, j \neq i) \tag{3.3}$$

$$\sum_{i,j\in S} Xij \leqslant -1 \quad (S \subset V, 2 \leqslant |S| \leqslant n-2) \tag{3.4}$$

$$Xij = 0 \quad or \quad 1 \quad (i,j) \leqslant A \tag{3.5}$$

Equations (3.1) and (3.2) show that each vertex j/i should be connected to another vertex i/j one once. Though, these two conditions are not enough to ensure that the result of the model has exactly one circuit. Therefore, equation (3.3) ensures that the final solution is not a sub-set of tours or a set of small tours. It should connect all vertices in only one route. While in equation (3.4) variable Xij well define and set to 1 if two vertices (i,j) are connected in the last tour graph, or 0 if not connected.

4. METHODOLOGY

The concept of LPB algorithm is inspired by the idea of accepting graduates from high school to university. Through the steps that are applied during the admission of learners, which are the

methods used to divide learners and group them according to their cumulative rate. Additionally, these methods are used to improve the behavior and level of performance of individuals after admission to the departments. Learners need to use new study habits because the methods they used to study in junior high do not work properly in college (Qayyum, A., 2018) (Chew, S.L., 2010) (Cao, L. and Nietfeld, J.L., 2007). However, this idea was used to form the optimization algorithm LPB. It has been designed based on the exploration and the exploitation phases. Where exploration means the algorithm finds new solutions in new regions, whereas exploitation refers to utilizing current solutions and refining them such that its fitness improves. Fig.3. shows both phases of the LPB algorithm.

In this algorithm as a first step within the exploitation phase, a group of individuals is selected from the population. These individuals are then divided into subgroups, and the best individuals are then selected from the subgroups depending on their fitness. As an exploration phase, their behavior and performance are then improved by having them work in groups. Where teamwork will provide information sharing among themselves when they study (crossover), this method will affect their behavior randomly (mutation). LPB uses crossover and mutation techniques of the genetic algorithm (Salih and Rashid, 2022). Visual 1 shows the Pseudo-code for LPB. Fig. 4, Fig. 5 Fig. 6 show the flowchart of LPB. Table 1 shows the algorithm LPB symbols definitions.

4.1 Crossover

It consists of switching between opposite values of the two parent chromosomes for the purpose of forming a new chromosome, and it is likely that the new chromosome will give a better solution than the solutions of the chromosomes from which it is formed. There are several types of commutative crossover, including one-point, two-point crossover, and other types (Hassanat, A., Almohammadi, 2019).

4.2 Mutation

In this process, specific genes within a single chromosome are altered or switched to form chromosomes that give new solutions to the subsequent generation. In order to obtain the best possible solutions, represented by the formation of the largest number of different chromosomes within the generation, there are several types: uniform mutation, replacement mutation, dynamic mutation (Hassanat, A., Almohammadi, 2019).

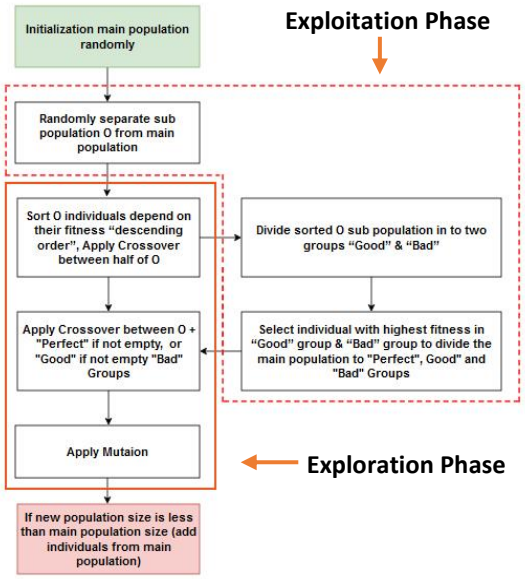


Fig. (3): The both phases of LPB Algorithm

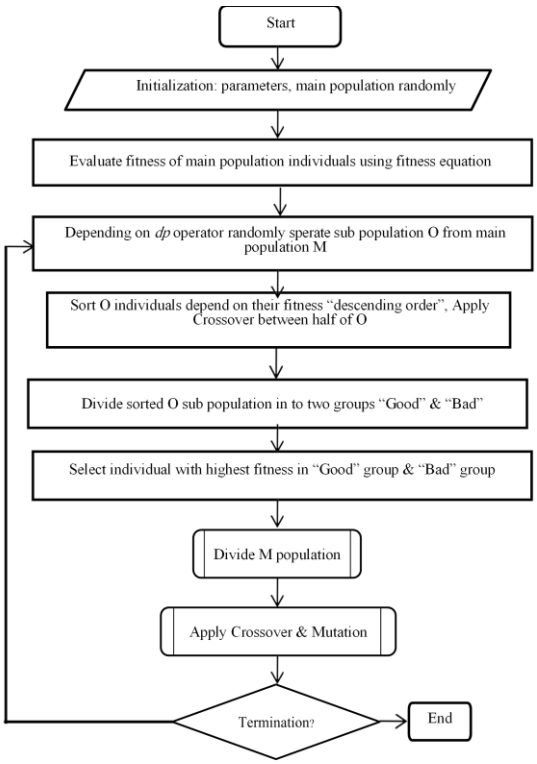


Fig. (4): LPB Algorithm Flowchart

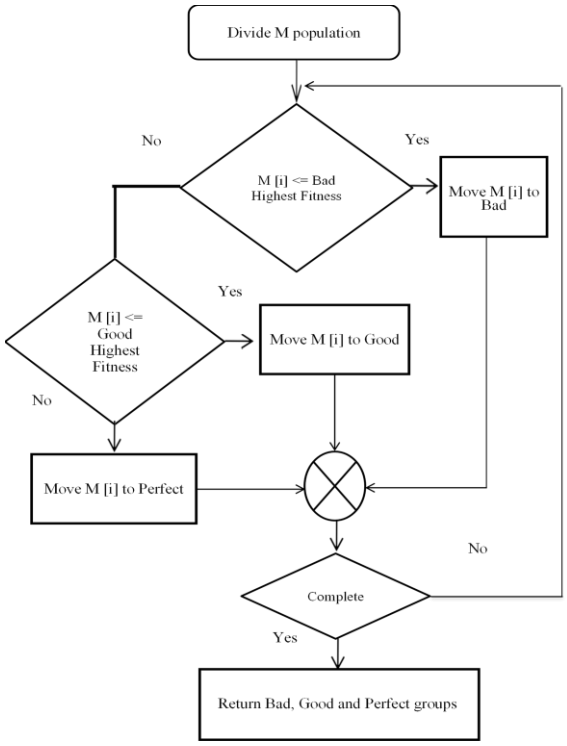


Fig (5): Divide M Population Flowchart

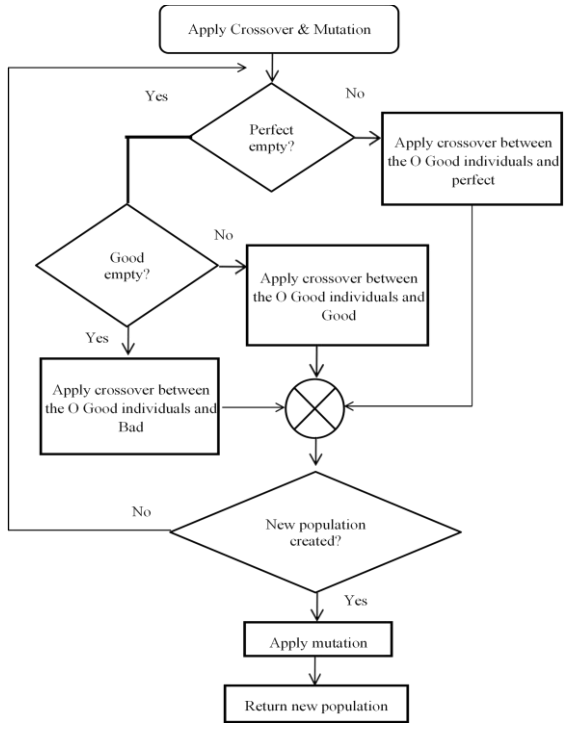


Fig. (6): Apply Crossover & Mutation

1. Randomly Initial a main population: M
2. Calculate fitness of all M individuals
3. Initial the parameters: the number of required individuals N, Crossover and mutation rates
4. Use *dp* parameter to randomly create sub population O from M
5. Sort the individuals in O (descending order) depending on their fitness
6. Divide O to two populations, good (Highest fitness) and bad (Lowest fitness)
7. Apply Crossover between half of O individuals
8. **While** termination condition is not met
9. Find the highest fitness in good and bad populations
   if an individual from M has fitness ≤ highest fitness in the bad population
   Move M individual to the bad population (BP)
   else if an individual from M has fitness ≤ highest fitness in the good population
   Move M individual to the good population (GP)
   else
   Move M individual to the perfect population (PF)
   end if
   while k ≤ N
   if PF is not empty
   Select an individual from PF
   Apply Crossover between PF and O Good individuals
   else if GP is not empty
   Select an individual from GP
   Apply Crossover between GP and O Good individuals
   else
   Select an individual from BP
   Apply Crossover between BP and O Good individuals
   end if
   k = k+1;
   end while
10. Mutation
11. **If** number of new population < number of main population
    Add individuals to from main population to new population
    end if
12. Termination
13. Repeat the procedure from step 4 until the termination condition is met.
14. **end while**
15. Optimal Solution: Select the best solution from the perfect population
    if an individual from M has fitness ≤ highest fitness in the bad population
    Move M individual to the bad population (BP)
    else if an individual from M has fitness ≤ highest fitness in the good population
    Move M individual to the good population (GP)
    else
    Move M individual to the perfect population (PF)
    end if

Visual 1: Pseudocode LPB

TABLE (1): LPB Symbols Definitions

Symbols	Meaning
M	Initial Random Population
N	Number of New Population Individuals
dp	Percentage Selected from M
O	Sub Population Selected from M Based on dp
k	Counter (Count the Number of New Created Individuals)
PF	Perfect Population
GP	Good Population
BP	Bad Population

5. SIMULATION EXPERIMENTS

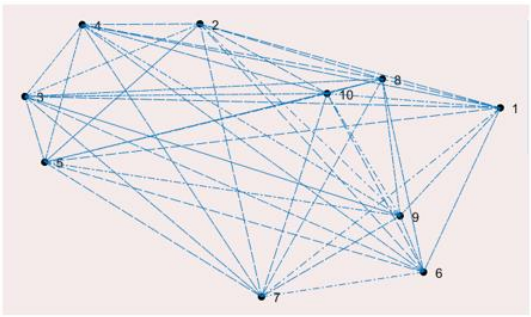
In this work, two groups of cities (10 cites) and (20 cites) were tested using LPB to solve TSP.

1) The following parameters are the ten cities' coordinates:

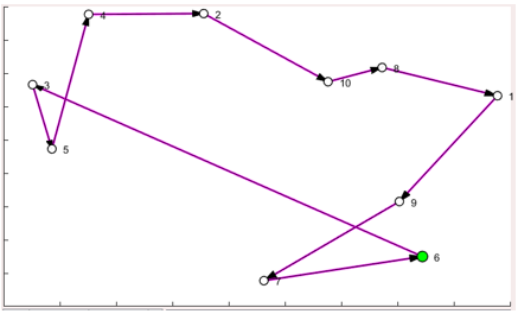
(87.6282, 63.2406), (35.5089, 87.8986), (5.1502, 66.5728), (15.0916, 87.6659), (8.5827, 47.2842), (74.3121, 14.9923), (46.1960, 7.7453), (67.1567, 71.7100), (70.2254, 31.4906), (57.5644, 67.4452).

TABLE (2): LPB APPLIED TO TSP FOR 10 CITES, RUN 10 ITERATIONS, POPULATION SIZE (50)

Iterations	Best Cost
1	410.9483
2	410.9483
3	410.9483
4	407.3445
5	353.7346
6	353.7346
7	342.6177
8	328.9473
9	328.9473
10	328.9473



(a) The Possible Paths



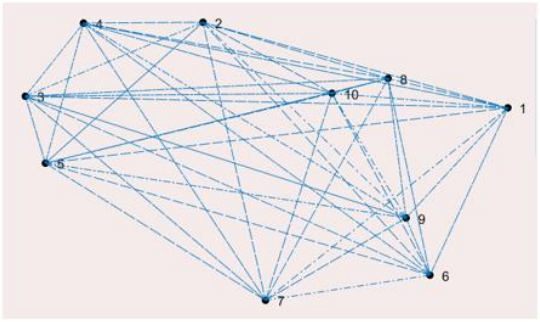
(b) The Best Path in 10th Generation (Iterations)

Fig. (7): The results of applying LPB on TSP for 10 cities in 10 iterations. Path Cost = 328.9473

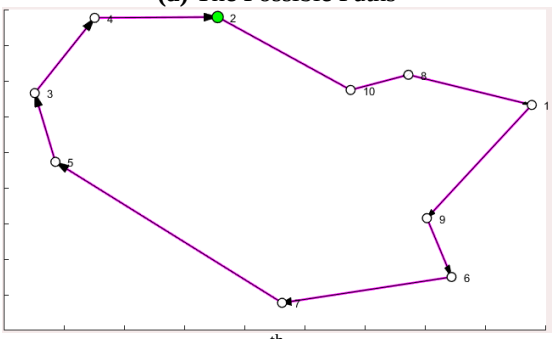
Table 2 and Fig. 7 presents the result of LPB that applied to TSP for 10 cites that run 10 iterations with a population size of (50).
Table 3 and Fig. 8 presents the result of LPB that applied to TSP for 10 cites that run 20 iterations with a population size of (50).

TABLE (3): LPB APPLIED TO TSP FOR 10 CITES, RUN 20 ITERATIONS, POPULATION SIZE (50)

Iterations	Best Cost
1	352.1032
2	352.1032
3	352.1032
4	314.3505
5	314.3505
6	297.6153
7	262.8710
8	262.8710
8	262.8710
9	262.8710
10	262.8710
11	262.8710
12	262.8710
13	262.8710
14	262.8710
15	262.8710
16	262.8710
17	262.8710
18	262.8710
19	262.8710
20	262.8710



(a) The Possible Paths



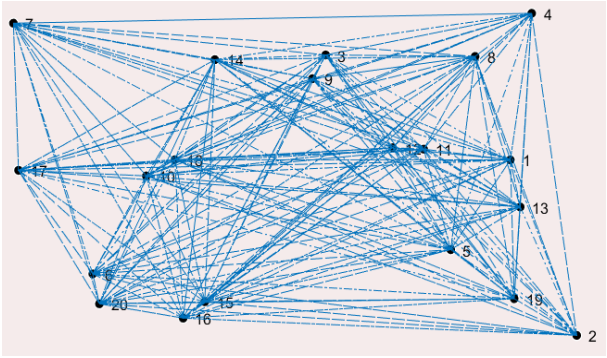
(b) The Best Path in 20th Generation (Iterations)

Fig. (8): The results of applying LPB on TSP for 10 cities in 20 iterations. Path Cost = 262.8710

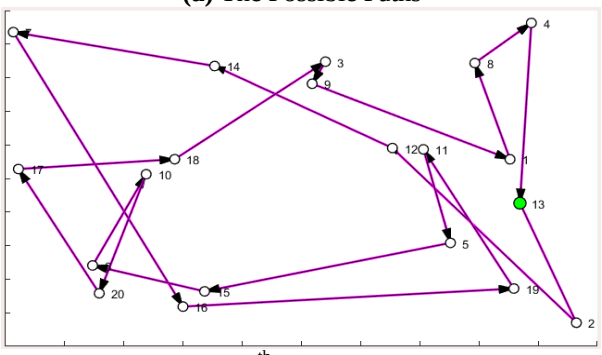
2) The following parameters are the Twenty cities' coordinates:
(85.2763, 55.6971), (96.4688, 7.0385), (54.1310, 84.7756), (88.8741, 96.3164), (75.2214, 30.8088), (14.8344, 24.0881), (1.4551, 93.5747), (79.3342, 84.3726), (51.8853, 78.1993), (23.8725, 51.2524), (70.6591, 58.6617), (65.4240, 59.0598), (86.9477, 42.6323), (35.4301, 83.4757), (33.7340, 16.3689), (30.0831, 11.8256), (2.3203, 52.7999), (28.6886, 55.7291), (8 .9246, 17.2080).

TABLE (4): LPB APPLIED TO TSP FOR 20 CITES, RUN 10 ITERATIONS, POPULATION SIZE (50)

Iterations	Best Cost
1	896.9424
2	875.9185
3	875.9185
4	875.9185
5	875.9185
6	811.2756
7	811.2756
8	766.0628
9	766.0628
10	766.0628



(a) The Possible Paths



(b) Best Path 10th Generation (Iterations)

Fig. (9) The results of applying LPB on TSP for 20 cities in 10 iterations. Path Cost = 766.0628

Table 4 and Fig. 9 present the result of LPB that applied to TSP for 20 cites, run 10 iterations with population size of (50).
Table 5 and Fig. 10 present the result of LPB applied to TSP for 20 cites, run 20 iterations.

TABLE (5): LPB APPLIED TO TSP FOR 20 CITES, RUN 20 ITERATIONS, POPULATION SIZE (50)

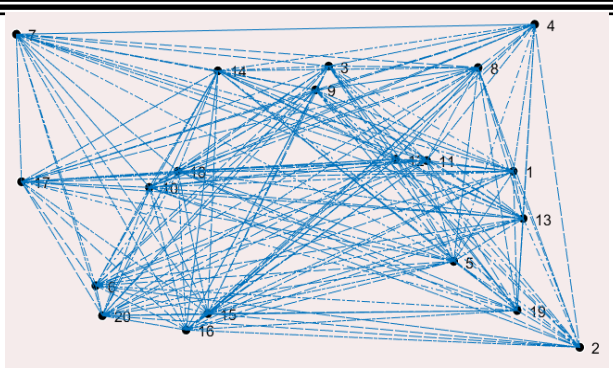
Iterations	Best Cost
1	855.9899
2	855.9899
3	855.9899
4	855.9899
5	849.4377
6	829.1420
7	821.0929
8	793.3323
9	793.3323
10	781.5133
11	775.0352
12	764.3390
13	759.1424
14	740.5747
15	728.5022
15	728.5022
16	728.5022
17	720.9793
18	714.2379
19	707.3087
20	707.3087

410.9483

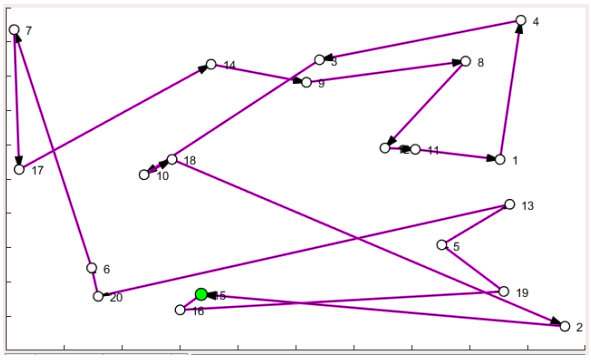
407.3445

407.3445

353.7346



(a) The Possible Paths



(b) Best Path 20th Generation (Iterations)

Fig. (10): The results of applying LPB on TSP for 20 cities in 20 iterations. Path Cost = 707.3087

As shown in Fig. 7, Fig. 8, Fig. 9 and Fig. 10 the LPB algorithm is implemented in a different number of cities and iterations, the algorithm proved that it is still able to solve the TSP problem as well as it did not generate sub tours.

6. Results and Discussion

The LPB algorithm has been examined under the same conditions for the experiment, and changes in the number of iterations or nodes. Another test for LPB was done by implementing on the Traveling Salesman Problem. Where, the number of cities (node) N = 30, the size of population pop = 100, and the algorithm runs (Iterations) = 500. Same parameters were applied on GA and PSO to compare between the algorithm's performance. As shown in Table 6, the results of the comparison. As it has been recorded in the run time 1, the minimum best cost was achieved by LPB (663.3729) against GA (705.4032) and PSO (766.3137). Same in run time 5 – 8 LPB best cost were (650.5823, 529.9301, 580.9361, 557.1948) while GA (799.7458, 652.5311, 752.6833, 664.4397) and PSO (744.9497, 535.9376, 808.2262, 722.5033). Same for most cases.

Table 7, shows the average and standard deviation of the three applied algorithms. Under the same experimental conditions, we can see that the LPB can find the optimal solution when there are a large of nodes, whereas the GA and PSO algorithms are stuck in the local optimal solution. It is evident that jumping out of the local optimal solution is simpler with the LPB. The minimum average of all run times was (655.1900067) for LPB which is better than GA (723.084896) and PSO (692.6886233). Fig. 11 shows the comparison between the LPB, GA and PSO algorithms applied on TSP.

TABLE (6): LPB, GA, PSO ON TSP RESULTS FOR 30 RUN TIMES (THE NUMBER OF ITERATIONS (ITER.) = 500)

un Time	Best Cost		
	LPB	GA	PSO
1 (500)	**663.3729**	705.4032	766.3137
2 (500)	677.4575	748.7583	**583.006**
3 (500)	746.4851	**646.082**	680.4097
4 (500)	641.7487	**632.6139**	666.4317
5 (500)	**650.5823**	799.7458	744.9497
6 (500)	**529.9301**	652.5311	535.9376
7 (500)	**580.9361**	752.6833	808.2262
8 (500)	**557.1948**	664.4397	722.5033
9 (500)	640.8687	750.1137	**663.8623**
10 (500)	**642.5863**	794.194	774.0129
11 (500)	**633.3348**	679.8234	705.8168
12 (500)	693.2512	820.9702	**648.2983**
13 (500)	802.4168	778.1102	**607.3172**
14 (500)	**666.3718**	733.8848	742.2518
15 (500)	**539.6779**	771.9345	626.7483
16 (500)	677.1764	656.8408	**649.2998**
17 (500)	**717.5688**	744.03138	815.4449
18 (500)	**697.9374**	796.6006	746.9769
19 (500)	723.5534	822.6372	**617.5658**
20 (500)	**620.1271**	766.404	735.505
21 (500)	591.1555	719.8724	**586.9506**
22 (500)	**561.1329**	803.1955	633.9511
23 (500)	744.7153	752.0911	**702.4987**
24 (500)	650.5992	**610.4039**	688.741
25 (500)	**527.6221**	676.6138	828.162
26 (500)	839.2412	770.6227	**708.1551**
27 (500)	620.3066	**576.6344**	803.2493
28 (500)	719.5151	**604.1562**	646.2788
29 (500)	732.0537	790.7508	**629.0284**
30 (500)	**566.7805**	670.404	712.7658

10 (500)	**642.5863**	794.194	774.0129
11 (500)	**633.3348**	679.8234	705.8168
12 (500)	693.2512	820.9702	**648.2983**
13 (500)	802.4168	778.1102	**607.3172**
14 (500)	**666.3718**	733.8848	742.2518
15 (500)	**539.6779**	771.9345	626.7483
16 (500)	677.1764	656.8408	**649.2998**
17 (500)	**717.5688**	744.03138	815.4449
18 (500)	**697.9374**	796.6006	746.9769
19 (500)	723.5534	822.6372	**617.5658**
20 (500)	**620.1271**	766.404	735.505
21 (500)	591.1555	719.8724	**586.9506**
22 (500)	**561.1329**	803.1955	633.9511
23 (500)	744.7153	752.0911	**702.4987**
24 (500)	650.5992	**610.4039**	688.741
25 (500)	**527.6221**	676.6138	828.162
26 (500)	839.2412	770.6227	**708.1551**
27 (500)	620.3066	**576.6344**	803.2493
28 (500)	719.5151	**604.1562**	646.2788
29 (500)	732.0537	790.7508	**629.0284**
30 (500)	**566.7805**	670.404	712.7658

TABLE (7): The average & standard deviation of LPB, GA and PSO

Algorithm	Average of 30 Run Times	Standard Deviation
LPB	**655.1900067**	77.38548213
GA	723.084896	68.89060456
PSO	692.6886233	73.7235773

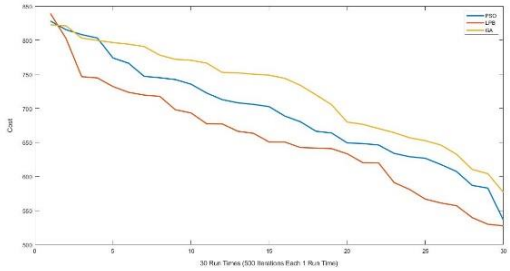


Fig. (11): Comparison the results of LPB, GA, and PSO algorithms on TSP

7. CONCLUSION

In this paper, we described the problem of the Traveling Salesman and its importance and the areas in which it is used. It's one of the most difficult and complex computing applications. This is because it is based on connecting a group of cities to get the shortest tour between them. Since each city will be toured only once, then return to the starting point. It has previously

been solved with several optimization algorithms. Thus, we have implemented for the first time by a new optimization algorithm called Learner performance-based behavior (LPB). LPB algorithm is one of the recent algorithms that is efficient in finding approximate optimal solutions to optimization problems. The idea of this algorithm came from the admission of students to universities according to their rates. All algorithms that take their idea from living reality are called metaheuristic algorithms. LPB Proved better results than (GA and PSO) optimization algorithms. Where they were carried out in 30 cities, the size of the population was 100, in 500 iterations. repeated 30 times.

REFERENCES

Gunantara, N. and Nurweda Putra, I., 2019. The characteristics of metaheuristic method in selection of path pairs on multicriteria ad hoc networks. *Journal of Computer Networks and Communications*, 2019.

Rahman, C. M., & Rashid, T. A. (2021). A new evolutionary algorithm: Learner performance-based behavior algorithm. Egyptian Informatics Journal, 22(2), 213-223.

Reinelt, G. (2003). The traveling salesman: computational solutions for TSP applications (Vol. 840). Springer.

Fogel, D. B. (1988). An evolutionary approach to the traveling salesman problem. Biological Cybernetics, 60(2), 139-144.

Cheng, C. B., & Mao, C. P. (2007). A modified ant colony system for solving the travelling salesman problem with time windows. Mathematical and Computer Modelling, 46(9-10), 1225-1235.

Hassanat, A.B., Prasath, V.S., Abbadi, M.A., Abu-Qdari, S.A. and Faris, H., 2018. An improved genetic algorithm with a new initialization mechanism based on regression techniques. Information, 9(7), p.167.

Fu, C., Zhang, L., Wang, X., & Qiao, L. (2018, May). Solving TSP problem with improved genetic algorithm. In AIP Conference Proceedings (Vol. 1967, No. 1, p. 040057). AIP Publishing LLC.

Kaabi, J. and Harrath, Y., 2019. Permutation rules and genetic algorithm to solve the traveling salesman problem. Arab journal of basic and applied sciences, 26(1), pp.283-291.

Hariyadi, P.M., Nguyen, P.T., Iswanto, I. and Sudrajat, D., 2020. Traveling salesman problem solution using genetic algorithm. Journal of Critical Reviews, 7(1), pp.56-61.

Khan, I., Pal, S. and Maiti, M.K., 2018, March. A modified particle swarm optimization algorithm for solving traveling salesman problem with imprecise cost matrix. In 2018 4th International Conference on Recent Advances in Information Technology (RAIT) (pp. 1-8). IEEE.

Yu, M., 2019. A solution of TSP based on the ant colony algorithm improved by particle swarm optimization. Discrete and Continuous Dynamical Systems-S, 12(4&5), pp.979-987.

Emambocus, B. A. S., Jasser, M. B., Hamzah, M., Mustapha, A., & Amphawan, A. (2021). An Enhanced Swap Sequence-Based Particle Swarm Optimization Algorithm to Solve TSP. IEEE Access, 9, 164820-164836.

Wei, B., Xing, Y., Xia, X., & Gui, L. (2021). A novel particle swarm optimization with genetic operator and its application to tsp. International Journal of Cognitive Informatics and Natural Intelligence (IJCINI), 15(4), 1-17.

Matai, R., Singh, S.P. and Mittal, M.L., 2010. Traveling salesman problem: an overview of applications, formulations, and solution approaches. Traveling salesman problem, theory and applications, 1.

Qayyum, A., 2018. Student help-seeking attitudes and behaviors in a digital era. International Journal of Educational Technology in Higher Education, 15(1), pp.1-16.

Chew, S.L., 2010. Improving classroom performance by challenging student misconceptions about learning. APS Observer, 23(4).

Cao, L. and Nietfeld, J.L., 2007. College Students' Metacognitive Awareness of Difficulties in Learning the Class Content Does Not Automatically Lead to Adjustment of Study Strategies. Australian Journal of Educational & Developmental Psychology, 7, pp.31-46.

Salih, S. A., & Rashid, T. A. (2022). Learner Performance-Based Behavior Optimization Algorithm: A Functional Case Study. In The International Conference on Innovations in Computing Research (pp. 467-482). Springer, Cham.

Hassanat, A., Almohammadi, K., Alkafaween, E.A., Abunawas, E., Hammouri, A. and Prasath, V.S., 2019. Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. Information, 10(12), p.390.