Discrete Particle Swarm Optimization for Travelling Salesman Problems: New Combinatorial Operators

Morad Bouzidi[1](✉), Mohammed Essaid Riffi[1], and Ahmed Serhir[2]

[1] LAROSERI Laboratory, Department of Computer Science, Chouaib Doukkali University, El Jadida, Morocco
mrbouzidi@gmail.com
[2] Department of Mathematics, Chouaib Doukkali University, El Jadida, Morocco

Abstract. The Particle Swarm Optimization is one of the most famous nature inspired algorithm that belongs to the swarm optimization family. It has already been used successfully in the continuous problem. However, this algorithm has not been explored enough for the discrete domain. In this work we introduce new operators that are dedicated to combinatorial research that we implemented on a modified discrete particle swarm optimization called DPSO-CO to solve travelling salesman problem. The experimental results on a set of different instances, and the comparison study with others adaptations show that adopting new ways, combinations and operators gives birth to a really competitive efficient algorithm in operational research.

Keywords: Combinatorial research · Discrete operators
Particle swarm optimization · Travelling salesman problem

1 Introduction

The metaheuristic algorithms, is a set of nature inspired methods, generally based on search probability technique to find a good solution within a reasonable time. Today, the metaheuristic algorithms have become more involved into solving a real engineer single and multi-objective problems; before applying the metaheuristics on a real problem, the researchers tested its quality performance to resolve classical problems. As far as the combinatorial problems is concerned, the travelling salesman problem (TSP) is considered as the most famous one that belongs to NP-hard problems [1], Its resolution consists in finding a short travel among a set of cities that the salesman has to visit in order to finish his journey at the initial starting point. Furthermore, TSP has several variations in different areas, which makes its resolution more attractive in computer wiring, vehicle routing and scheduling problems. Moreover, numerous metaheuristics have been published for TSP such as genetic algorithm [2], harmony search algorithm [3], particle swarm optimization [4], ant colony optimization [5] and bee colony optimization [6].

A. Abraham et al. (Eds.): SoCPaR 2017, AISC 737, pp. 141–150, 2018.
https://doi.org/10.1007/978-3-319-76357-6_14

The particle swarm optimization (PSO) [7] is one of the well known optimization algorithms, inspired by bird flocks intelligence method, and classified on swarm intelligence algorithm. The principle of PSO focuses on the collaboration of all particles swarm, where each particle seeks a good position and moves with a specific velocity. This algorithm defines a specific rule which allows each particle to follow the best position in swarm without ignoring its own search making it always search for what is best. In fact, several adaptations of PSO called discrete particle swarm optimization (DPSO) have been proposed [8–10] for a different combinatorial optimization problems, one of the most known adaptations was introduced by Clerc [11], but the experience showed that adaptation of DPSO does not give competitive results like other metaheuristics methods.

This work presents a new adaptation of DPSO named DPSO-CO with the introduction of new combinatorial operators that follows the idea of Clerc in addition to unification dimension, which is then concluded by the presentation of a performance DPSO. The rest of this paper is organized as follows: definition of travelling salesman problem in the Sect. 2 and description of particle swarm optimization algorithm in the Sect. 3, while the Sect. 4 is devoted to give presentation of new operators and a new adaptation. The application and the study of experimental results of this proposition comes in the Sect. 5, and the conclusion and perspective in the Sect. 6.

2 Travelling Salesman Problem

The travelling salesman problem consists on finding the shortest path for the salesman to take in order to visit all cities while two rules must be respected: first, he needs to visit each city only once, second, he has to finish at the starting point. Mathematically speaking, the solution of this problem is a permutation π, contains n elements, where n presents the number of cities, and the i^{th} element π_i indicate the i^{th} city that should be visited, therefore, the objective function to be minimized is:

$$f(x) = \sum_{i=1}^{n-1} distance(\pi_i, \pi_{i+1}) + distance(\pi_n, \pi_1) \quad (1)$$

3 Particle Swarm Optimization

In 1995, Kennedy and Eberhart introduced the particle swarm optimization [12] that is a research algorithm based on the cooperation and sharing information within a defined research space in order to find a good food source like the social behaviour in bird flocking and fish schooling for example In researcher space, the algorithm launches a set of individuals as candidate solution called “particles”, each particle has a position known by all the group. In every moment, particles are moving with a specific velocity towards the best position without ignoring their previous ones that will be shared in the group once found.

More concretely, in D-dimensional space, a population of n particles as potential solutions, each particle has a position P^t (Eq. 2) which is generated randomly in t = 0.

$$P^t = \left(X_i^t\right)_{i=1,..,n} = \left(x_{i1}^t, x_{i2}^t, \ldots, x_{iD}^t\right)_{i=1,..,n} \tag{2}$$

However, before next iteration t, each i th particle is based on the global best position founded in group $gbest^{t-1} = \left(gbest_k^{t-1}\right)_{k=1,..,D}$, and its best previous position $pbest_i^{t-1} = \left(pbest_{i,k}^{t-1}\right)_{k=1,..,D}$ to calculate its next velocity $V_i^t = \left(v_{i,k}^t\right)_{k=1,..,D}$ according Eq. 3, which will take it towards another position $X_i^t = \left(x_{i,k}^t\right)_{k=1,..,\mathrm{D}}$ by formulation of Eq. 4.

$$V_i^t = \omega \times V_i^{t-1} + c_1 \times r_1 \times \left(pbest_i^{t-1} - X_i^{t-1}\right) + c_2 \times r_2 \times \left(gbest^{t-1} - X_i^{t-1}\right) \tag{3}$$

$$X_i^t = X_i^{t-1} + V_i^t \tag{4}$$

Where ω is the inertia coefficient constant in interval [0, 1], c1 and c2 are cognitive and social parameters constants in interval [0, 2], and r_1, r_2 are two generated parameters randomly in [0, 1]. This process keeps repeating until the stopping condition will be reached. As in our case, the objective is to find the best cycle of TSP without exceeding a maximum iteration number, Algorithm 1 resumes this process with a pseudo code of PSO.

Algorithm 1. Pseudo code of proposed method

```
Begin
Initial population P with d solutions randomly
For each p in P do
  Initial their velocity : vp
  Initial their best position founded  : pbest
end For each
Get gbest : the best pbest in P
While ((optimum not finder) and (iteration number is not
archived)) do
  For each p in P do
    Calculate vp with equation 3
    Update p following equation 4
    if(cost(p)<cost(pbest)) then
      pbest = p
      if(cost(p)<cost(gbest)) then
        gbest = p
      end if
    end if
  end for each
end while
return gbest
End.
```

4 Proposed Particle Swarm Optimization with Discrete Operators

The proposed discrete particle swarm optimization with combinatorial operators (DPSO-CO), follows the same method of operators presented by Clerc [11], but they focus mainly on the unification of the dimension of velocity and position, in order to avoid any truncate of velocity so that no data would be lost. This section is devised in three sub-sections, where the first one contains a presentation of the novel operators while in the second one adaptation of DPSO for this operator is to be found.

4.1 Novel Discrete Operators

Before starting representing operators, as it is known, the structure of the position is represented as a permutation π, and velocity is a set of permutation. In this proposed operators, we defined the data structure of velocity like position, which means that the velocity is a permutation of d elements, where d is dimension of research space.

$$v = (v_k)_{k=1,..,d} \tag{5}$$

Addition for position and velocity. The result of addition between velocity and position is a position, which velocity translates the order of item position, that means in each i^{th} item of x will be the $(v_i)^{th}$ item in x' resumed in Eq. 6, to clarify this operation formulate 7 and Fig. 1. Example of position x plus velocity vshow example of addition between position and velocity.

$$x + v = x' = \left(x_i^{'}\right)_{i=1,..,\mathrm{d}} = (x_k)_{i=1,..,\mathrm{d}}/i = v_k \tag{6}$$

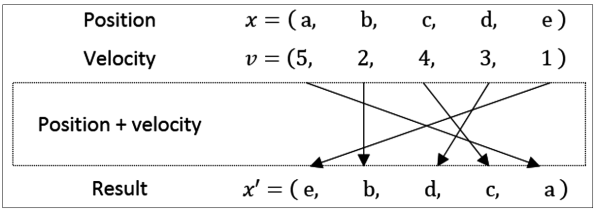

Fig. 1. Example of position x plus velocity v

$$\left.\begin{array}{l} x = (1,4,3,5,2) \\ v = (5,2,4,3,1) \end{array}\right\} x + v = (2,4,5,3,1) \tag{7}$$

With this operator, if addition of position and two velocities give the same position, it implies that they are equal (Eq. 8), and this redefinition makes more sense of vector translation to velocity.

$$x + v' = x + v'' \Rightarrow v' = v'' \tag{8}$$

Addition between two velocities. The addition of velocity v^1 with another velocity v^2 is a new velocity v^3 with d elements, where each item i in v^3 equal value of $(v_i^1)^{th}$ item in v^2, this action resumes to translation of v^1 by v^2, where $(x + v^1) + v^2 = x + (v^1 + v^2)$ but $v^1 + v^2$ not equals $v^2 + v^1$. Equation 9 resumes an example of this operation.

$$v^1 + v^2 = v^3 = \left(v_i^3\right)_{i=1,..,d} = \left(v^2_{\left(v_i^1\right)}\right)_{i=1,..,d} \tag{9}$$

$$\left.\begin{array}{l} v^1 = (2,5,1,4,3) \\ v^2 = (3,1,2,5,4) \end{array}\right\} v^3 = v^1 + v^2 = (1,4,3,5,2) \tag{10}$$

Subtraction operator. At the same idea of addition operator, it is applied between two positions, so the x^1 minus x^2 is v where each i^{th} item in v (v_i^{th}) equal the rang k flowing Eq. 11, the result is a velocity v which enables the second position to move towards the first position. And if x^1 - x^2 = v then x^1 = x^2 + v.

$$x^1 - x^2 = v = (v_i)_{i=1,..,\,d} = (k)_{i=1,..,d} / x_k^1 = x_i^2 \tag{11}$$

$$\left.\begin{array}{l} x^1 = (3,1,4,2,5) \\ x^2 = (1,4,3,5,2) \end{array}\right\} v = x^1 - x^2 = (2,3,1,5,4) \tag{12}$$

Multiplication operator. In this operator, the coefficient represents a probability parameter of adjustment for velocity, in other way, the particle moves with some velocity which can include some problems that can create a small disruptive impact on its position for the next iteration. Thus, the multiplication between a coefficient c and velocity v. After that the operator checks if a random number between 0 and 1 is less than c, then he applies a random swap in v, otherwise it does nothing (Eq. 13).

$$cv = \begin{cases} random\ swap\ of\ v & rand() < c \\ v & otherwise \end{cases} \tag{13}$$

4.2 Discrete Particle Swarm Optimization

The proposal method DPSO-CO based on these new defined operators generates randomly particles population, then starts the search loop, where it detects, in each iteration, g_{best} and generates the next solution for each particle following Eqs. 3 and 4, where the inertia coefficient constant takes value 1, c1 and c2 are generated for each iteration between 0 and 1. Each new solution gets a small perturbation with 2-opt in order to look for a better neighbour, 2-opt improves a solution by applying iteratively exchanges between two edges resuming in Fig. 2. 2-Opt Swap, where (A) represents the initial case, and (B) is the new perturbation of tour. In TSP case, to accept exchange

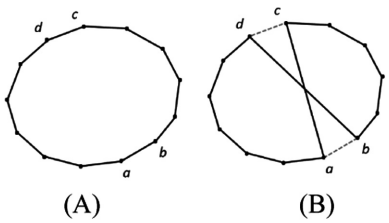

Fig. 2. 2-Opt Swap

between two pairs (a, b) and (c, d) to (a, c) and (b, d) the sum of distances of (a, c) and (b, d) must be less than the sum of pairs (a, b) and (d, c). This process continues until DPSO-CO finds the best known solution for the problem or if it exceeds the maximal number of possible iterations.

5 Experimental Results

In order to validate its performances, the proposed adaptation has been programmed with C++ language and has been made on PC with processor Intel(R) Core(TM) i5-2500 CPU @ 3.30 GHz and 4 Go of RAM, and has been tested on some symmetric benchmarks of TSPLIB library [13].

Table 1 represents the numerical results on twenty instances of symmetric problems, each instance is executed thirty times, the first column represents the name of instance problem, the second column Opt shows the best known optimum of instance, while the third and fourth columns show respectively the best and worst solutions obtained by DPSO-CO. In the fifth column the average length of all solutions is to be found, and in the next column the standard deviation SD of all the results is shown. The seventh and eighth columns represent respectively the percentage relative error of the average PDav and the best solution PDbest according to the optimal known solution presented in second column, this value is calculated following Eq. 14. Finally in the ninth column $C_{1\%}/C_{opt}$, where C1% indicates the number of solutions where relative error is less than 1 and Copt is the number of solutions equal to optimum known solution that means the number of iteration which its relative error is null, and the last one is time column that shows the average of time execution of all iterations in second.

$$PDsolution = \frac{Solution\ lenght - \ optimal\ known\ lenght}{optimal\ known\ lenght} \times 100\% \qquad (14)$$

The experimental results have shown that this proposed method gets the optimum one for the majority of instances, especially for medium-sized category instances where the average execution time is the shortest. To verify its performance according to the current DPSO, we have implemented DPSO presented by Clerc [11] and showed the results in Table 2. Comparison of the proposed algorithm with DPSO proposed by Clerc [11]. When comparing, DPSO gets the best known solutions for some instances, but in general, it does not provide results better than DPSO-CO that give solutions in a

Table 1. Results of the proposed method DPSO-CO for some symmetric instance of TSPLib.

Instance	Opt	Best	Worst	Average	SD	PDav	PDbest	$C_{1\%}/C_{opt}$	Time
eil51	426	**426**	427	426.07	0.25	0.02	**0.00**	30/28	0.22
berlin52	7,542	**7,542**	**7,542**	**7,542.00**	**0.00**	**0.00**	**0.00**	**30/30**	0.02
st70	675	**675**	**675**	**675.00**	**0.00**	**0.00**	**0.00**	**30/30**	0.13
eil76	538	**538**	540	538.53	0.72	0.01	**0.00**	30/18	2.09
pr76	108,159	**108,159**	**108,159**	**108,159.00**	**0.00**	**0.00**	**0.00**	**30/30**	0.11
rat99	1,211	**1,211**	1,212	1,211.07	0.25	0.01	**0.00**	30/28	2.29
kroA100	21,282	**21,282**	**21,282**	**21,282.00**	**0.00**	**0.00**	**0.00**	**30/30**	0.28
kroB100	22,141	**22,141**	**22,141**	**22,141.00**	**0.00**	**0.00**	**0.00**	**30/30**	1.63
kroD100	21,294	**21,294**	21,309	21,294.50	2.69	**0.00**	**0.00**	30/29	2.25
kroE100	22,068	**22,068**	22,106	22,069.40	6.85	0.01	**0.00**	30/28	3.93
rd100	7,910	**7,910**	7,911	79,10.07	0.25	**0.00**	**0.00**	30/28	2.19
eil101	629	**629**	635	631.20	1.78	0.35	**0.00**	30/06	6.49
lin105	14,379	**14,379**	**14,379**	**14,379.00**	**0.00**	**0.00**	**0.00**	**30/30**	0.27
pr107	44,303	**44,303**	44,387	44,309.50	18.43	0.01	**0.00**	30/26	5.48
ch130	6,110	**6,110**	6,147	61,22.57	11.44	0.21	**0.00**	30/08	15.79
pr136	96,772	**96,772**	97,105	96,870.70	86.98	1.10	**0.00**	30/07	18.27
pr144	58,537	**58,537**	**58,537**	**58,537.00**	**0.00**	**0.00**	**0.00**	**30/30**	0.48
Ch150	6,528	**6,528**	6,561	6,537.37	10.91	0.14	**0.00**	30/16	20.45
kroA150	26,524	**26,524**	26,689	26,556.20	41.66	0,12	**0.00**	30/03	27.18
kroB150	26,130	**26,130**	26,237	26,152.70	23.61	0,09	**0.00**	30/04	28.01
rat195	2,323	2,336	2,370	2,355.70	7.20	1,41	0.56	02/00	65.60
kroA200	29,368	**29,368**	29,599	29,495.30	55.63	0,43	**0.00**	30/01	83.45
ts225	126,643	**126,643**	**126,643**	**126,643.00**	**0,00**	**0,00**	**0.00**	**30/30**	0.86
tsp225	3,916	3,939	3,983	3,964.90	12.18	1.25	0.59	02/00	64.67

shorter time and with less amount of relative errors terms of who get result in small time and with a less reduced amount of relative errors. Both Fig. 3. Comparison average relative error between proposed method and PSO-ACO-3Opt [14] and Fig. 4. Comparison average execution time between proposed method and PSO-ACO-3Opt [14] show the difference of the PDav and the execution time between the proposed DPSO and the latest improved hybrid DPSO with Ant colony and 3-opt called PSO-ACO-3opt [14], from these figures DPSO-CO is more efficient than PSO-ACO-3Opt in terms of time and quality.

In addition, Table 3. Comparison of the proposed algorithm with DCS [15]. compares results of DPSO-CO with DCS [15], where column Time represents the average time for all iteration average execution, and the last line represents the average of each column. The results show that their performances are very close, and DPSO-CO gives a nice result especially in time of execution for medium-size instances, but when the size of the problem increases, DCS takes the advantage.

Table 2. Comparison of the proposed algorithm with DPSO proposed by Clerc [11].

Instance	DPSO				DPSO-CO			
	PDbest	PDav	$C_{1\%}/C_{Opt}$	Time	PDbest	PDav	$C_{1\%}/C_{Opt}$	Time
eil51	**0.00**	0.14	**30**/13	2.12	**0.00**	0.02	**30**/28	0.22
st70	**0.00**	0.01	**30**/29	4.82	**0.00**	**0.00**	**30/30**	0.13
eil76	**0.00**	0.94	15/03	7.87	**0.00**	0.01	**30**/18	2.09
pr76	**0.00**	0.04	**30**/15	8.38	**0.00**	**0.00**	**30/30**	0.11
kroB100	**0.00**	0.17	**30**/06	21.57	**0.00**	**0.00**	**30/30**	1.63
kroD100	**0.00**	0.38	**30**/02	21.18	**0.00**	**0.00**	**30**/29	2.25
kroE100	0.13	0.40	**30**/00	21.84	**0.00**	0.01	**30**/28	3.93
eil101	0.79	1.84	02/00	20.80	**0.00**	0.35	**30**/06	6.49
lin105	**0.00**	**0.00**	**30/30**	20.89	**0.00**	**0.00**	**30/30**	0.27
pr107	**0.00**	0.21	**30**/01	22.81	**0.00**	0.01	**30**/26	5.48
ch130	0.20	0.90	19/00	51.93	**0.00**	0.21	**30**/08	15.79
pr136	0.21	0.76	26/0	44.14	**0.00**	1.10	**30**/07	18.27
pr144	**0.00**	**0.00**	**30/30**	31.79	**0.00**	**0.00**	**30/30**	0.48
Ch150	0.34	1.08	12/00	64.49	**0.00**	0.14	**30**/16	20.45
kroA150	0.36	1.04	12/00	71.22	**0.00**	0.12	**30**/03	27.18
kroB150	0.24	0.86	24/00	68.39	**0.00**	0.09	**30**/04	28.01
rat195	2.28	3.37	00/00	145.70	0.56	1.41	02/00	65.60
kroA200	0.85	1.31	05/00	184.59	**0.00**	0.43	**30**/01	83.45
ts225	0.07	0.31	30/0	216.03	**0.00**	**0.00**	**30/30**	0.86

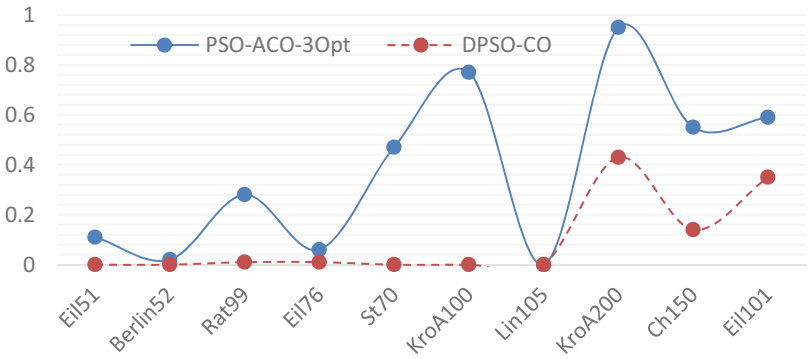

Fig. 3. Comparison average relative error between proposed method and PSO-ACO-3Opt [14].

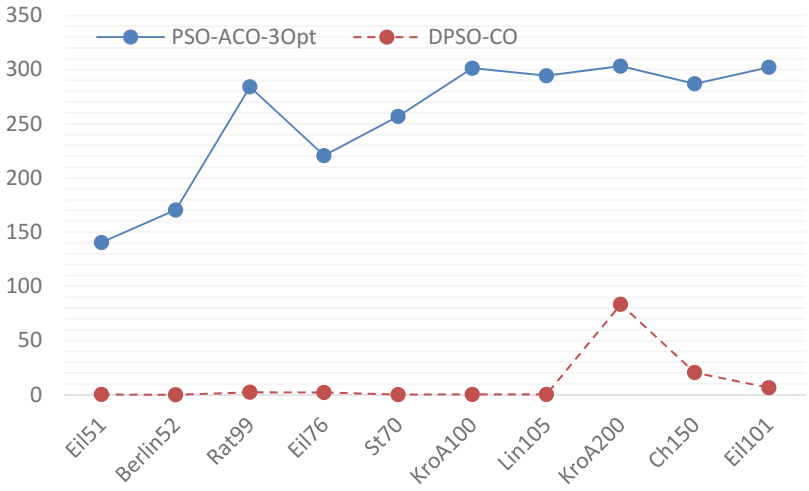

Fig. 4. Comparison average execution time between proposed method and PSO-ACO-3Opt [14].

Table 3. Comparison of the proposed algorithm with DCS [15].

Instance	PDbest		PDav		$C_{1\%}/C_{Opt}$		Time	
	DCS	DPSO-CO	DCS	DPSO-CO	DCS	DPSO-CO	DCS	DPSO-CO
eil51	**0.00**	**0.00**	**0.00**	0.02	**30/30**	**30**/28	1.16	0.22
st70	**0.00**	**0.00**	**0.00**	**0.00**	**30/30**	**30/30**	1.56	0.13
eil76	**0.00**	**0.00**	**0.00**	0.01	**30**/29	**30**/18	6.54	2.09
pr76	**0.00**	**0.00**	**0.00**	**0.00**	**30/30**	**30/30**	4.73	0.11
kroB100	**0.00**	**0.00**	**0.00**	**0.00**	**30**/29	**30/30**	8.74	1.63
kroD100	**0.00**	**0.00**	**0.04**	**0.00**	**30**/19	**30**/29	8.74	2.25
kroE100	**0.00**	**0.00**	**0.00**	0.01	**30**/18	**30**/28	14.18	3.93
eil101	**0.00**	**0.00**	0.22	0.35	**30**/06	**30**/06	18.74	6.49
lin105	**0.00**	**0.00**	**0.00**	**0.00**	**30/30**	**30/30**	5.01	0.27
pr107	**0.00**	**0.00**	**0.00**	0.01	**30**/27	**30**/26	12.89	5.48
ch130	**0.00**	**0.00**	0.42	0.21	28/07	**30**/08	23.12	15.79
pr136	0.01	**0.00**	0.24	1.10	**30**/00	**30**/07	35.82	18.27
pr144	**0.00**	**0.00**	**0.00**	**0.00**	**30/30**	**30/30**	2.96	0.48
Ch150	**0.00**	**0.00**	**0.33**	0.14	29/10	**30**/16	27.74	20.45
kroA150	**0.00**	**0.00**	0.17	0.12	**30**/07	**30**/03	31.23	27.18
kroB150	**0.00**	**0.00**	0.11	0.09	**30**/05	**30**/04	33.01	28.01
rat195	0.04	0.56	0.81	1.41	20/00	02/00	57.25	65.60
kroA200	0.04	**0.00**	0.26	0.43	29/00	**30**/01	62.08	83.45
ts225	**0.00**	**0.00**	0.01	**0.00**	**30**/26	**30/30**	47.51	0.86
Average	0.01	0.01	0.14	0.21	29/18	29/19	21.21	14.89

6 Conclusion

This paper presents a new adaptation of DPSO-CO characterized by a novel definition of discrete operators. This proposed DPSO-CO method has been applied on different symmetric TSP instances from TSPLib. The result was compared in confrontation with the last hybrid PSO known and a recent competitive algorithm DCS. From this obtained study, it can be concluded that the proposed DPSO-CO are effective and more performant than other methods. However, this work opens new horizons for DPSO-CO to solve other combinatorial problems especially when using open discrete operators in new different ways that can be used in the future for others metaheuristic algorithms to test and increase the performance of research.

References

1. Arora, S.: Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. J. ACM (JACM) **45**, 753–782 (1998)
2. Grefenstette, J., Gopal, R., Rosmaita, B., Van Gucht, D.: Genetic algorithms for the traveling salesman problem. In: Proceedings of the First International Conference on Genetic Algorithms and their Applications. Lawrence Erlbaum, Hillsdale (1985)
3. Bouzidi, M., Riffi, M.E.: Adaptation of the harmony search algorithm to solve the travelling salesman problem. J. Theor. Appl. Inf. Technol. **62**(1) (2014)
4. Wang, K.P., Huang, L., Zhou, C.G., Pang, W.: Particle swarm optimization for traveling salesman problem. In: 2003 International Conference on Machine Learning and Cybernetics (2003)
5. Dorigo, M., Birattari, M.: Ant colony optimization. In: Encyclopedia of Machine Learning. Springer (2010)
6. Wong, L.P., Low, M.Y.H., Chong, C.S.: A bee colony optimization algorithm for traveling salesman problem. In: Modeling & Simulation, AICMS 2008 (2008)
7. Kennedy, J.: Particle swarm optimization. In: Encyclopedia of Machine Learning, pp. 760-766. Springer, US (2011)
8. Chen, A.L., Yang, G.K., Wu, Z.M.: Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. J. Zhejiang Univ.-Sci. A **7**(4), 607–614 (2006)
9. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation (1997)
10. Pan, Q.K., Tasgetiren, M.F., Liang, Y.C.: A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Comput. Oper. Res. **35**(9), 2807–2839 (2008)
11. Clerc, M.: Discrete particle swarm optimization, illustrated by the traveling salesman problem. In: New Optimization Techniques in Engineering, pp. 219–239 (2004)
12. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science (1995)
13. Reinelt, G.: TSPLIB—A traveling salesman problem library. ORSA J. Comput. **3**(4), 376–384 (1991)
14. Mahi, M., Baykan, Ö.K., Kodaz, H.: A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem. Appl. Soft Comput. **30**, 484–490 (2015)
15. Ouaarab, A., Ahiod, B., Yang, X.S.: Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput. Appl. **24**(7–8), 1659–1669 (2014)