

An improved fruit fly optimization algorithm for solving traveling salesman problem^{*}

Lan HUANG^{†1,2}, Gui-chao WANG^{†1,2}, Tian BAI^{1,2}, Zhe WANG^{†‡1,2}

⁽¹⁾College of Computer Science and Technology, Jilin University, Changchun 130012, China

⁽²⁾Key Laboratory of Symbolic Computation and Knowledge Engineering (Jilin University), Ministry of Education, Changchun 130012, China

[†]E-mail: huanglan@jlu.edu.cn; wgc290029@163.com; wz2000@jlu.edu.cn

Received June 22, 2016; Revision accepted Jan. 23, 2017; Crosschecked Nov. 6, 2017

Abstract: The traveling salesman problem (TSP), a typical non-deterministic polynomial (NP) hard problem, has been used in many engineering applications. As a new swarm-intelligence optimization algorithm, the fruit fly optimization algorithm (FOA) is used to solve TSP, since it has the advantages of being easy to understand and having a simple implementation. However, it has problems, including a slow convergence rate for the algorithm, easily falling into the local optimum, and an insufficient optimization precision. To address TSP effectively, three improvements are proposed in this paper to improve FOA. First, the vision search process is reinforced in the foraging behavior of fruit flies to improve the convergence rate of FOA. Second, an elimination mechanism is added to FOA to increase the diversity. Third, a reverse operator and a multiplication operator are proposed. They are performed on the solution sequence in the fruit fly's smell search and vision search processes, respectively. In the experiment, 10 benchmarks selected from TSPLIB are tested. The results show that the improved FOA outperforms other alternatives in terms of the convergence rate and precision.

Key words: Traveling salesman problem; Fruit fly optimization algorithm; Elimination mechanism; Vision search; Operator
<https://doi.org/10.1631/FITEE.1601364>

CLC number: TP181

1 Introduction

The traveling salesman problem (TSP), a classic combinatorial optimization problem, is applied in many engineering applications, such as computer networking, hardware design, traffic route design, electronic control systems, and asynchronous transfer mode (ATM) packet-switched networks. Additionally, it is a typical non-deterministic polynomial (NP) hard problem, though the solution space for one TSP including n cities is $n!$.


TSPs can be easy to describe but difficult to

solve. The ways to solve the problem can be divided mainly into two categories: One is the exact method which can ensure obtaining the optimal solution, such as the branch-and-bound method (Lawler and Wood, 1966) and dynamic programming (Bellman and Dreyfus, 1962). However, due to the increase in the number of cities, the execution time for these methods expands exponentially (Little *et al.*, 1963). Thus, exact methods are suitable only for solving small-scale problems. The other is the approximate method which cannot guarantee obtaining the optimal solution but take less time, such as the genetic algorithm (GA), artificial ant colony (ACO) algorithm, particle swarm optimization (PSO) algorithm, and simulated annealing (SA) algorithm.

Grefenstette *et al.* (1985) proposed GA for TSP. To solve large-scale TSPs, Ding *et al.* (2007) presented an adaptive two-level GA. Liu and Zeng

[‡] Corresponding author

^{*} Project supported by the National Natural Science Foundation of China (Nos. 61472159 and 61373051)

 ORCID: Lan HUANG, <http://orcid.org/0000-0003-3223-3777>

© Zhejiang University and Springer-Verlag GmbH Germany 2017

(2009) proposed a GA with reinforcement learning to solve TSP. The algorithm uses GA as the framework and reinforcement learning as a mutation operator, and increases the convergence rate of GA. Dorigo and Gambardella (1997) presented an ACO algorithm which is capable of solving TSP. Escario *et al.* (2015) extended the ACO algorithm to solve TSP. The self-organizing dynamics of a real ant population was used to provide adaptive capacities to the ACO algorithm for TSP. Hendtlass (2003), Clerc (2004), and Hoffmann *et al.* (2011) developed PSO algorithms with some modifications or hybrid-learning schemes to solve TSP. Mahi *et al.* (2015) proposed a new hybrid method based on the PSO, ACO, and 3-opt algorithms to solve TSP. In this method, PSO is used for determining parameters that affect the performance of ACO, and the 3-opt is used for getting rid of the local solution found in the ACO algorithm. Kirkpatrick (1984) first applied the SA algorithm to deal with TSP. Geng *et al.* (2011) mixed an effective local search algorithm with the SA algorithm and greedy search techniques to solve TSP. Karaboga and Gorkemli (2011) proposed a combinatorial artificial bee colony (ABC) algorithm and applied it to solve TSP. They proved that the ABC algorithm can also be used to solve combinatorial optimization problems. Ouyang *et al.* (2013) proposed a novel discrete cuckoo search algorithm for a spherical TSP. Zhou *et al.* (2015) proposed a discrete invasive weed optimization algorithm to solve TSP. They mixed a 3-opt local search operator and a 2-opt local search operator with an invasive weed optimization algorithm to search for optimal results. Jolai and Ghanbari (2010) presented an improved artificial neural network (ANN) approach to solve TSP. They improved the accuracy of the results and obtained the optimal tours with smaller total distances with a Hopfield neural network and data transformation techniques.

Although these algorithms have improved the precision of the optimal result of TSP, they still run easily into problems at the local optimum and hardly meet the requirement of focusing on precision. After studying a variety of swarm intelligence algorithms and evaluating their test results, we find that FOA not only is easy to understand and simple to implement, but also has a higher expanding ability in its smell search process. However, the vision search process

always keeps the algorithm trapped in the local optimum.

To overcome these limitations, in this study, an elimination-based fruit fly optimization algorithm (EFOA) is proposed based on FOA to solve TSP. First, compared with FOA, EFOA reinforces the vision search process. In the improved vision search process, other fruit flies constantly observe the location of the local optimal fruit fly and approach it gradually, but not by just rapidly and directly flying to the position of the best individual. Furthermore, an elimination mechanism is added to FOA. It means that some poor individuals of the group will be eliminated and some new fruit fly individuals will be generated. The elimination mechanism adds diversity to the population to improve the expanding ability and maintain the convergence ability. Finally, EFOA is used to solve TSP. In the process, two operators, an improved reverse operator and a new multiplication operator, are proposed. Compared with the other algorithms, EFOA has a better performance when testing the benchmark data selected from TSPLIB, and yields a higher precision.

2 Fruit fly optimization algorithm

FOA is a new swarm-intelligence optimization algorithm proposed by Pan (2011; 2012). It depicts the foraging behavior of the fruit fly. Due to their keen sense of smell and vision, fruit flies can judge the general direction to find food very quickly according to odor concentrations in the air in the process of foraging. Within the scope of their proximity to food, they constantly approach those partners closest to food using their visual sensitivity. Pan (2011; 2012) summed up the foraging behavior of fruit flies as a random search process and visual localization process, and then proposed FOA. Compared with other swarm intelligence algorithms (such as PSO, ACO, and ABC), FOA has the advantages of being easy to understand and having a simple implementation. The specific steps are described as follows:

Step 1: initialization. Define the group size as P , the maximum number of algorithm iteration as m , and the original location of group as (X_0, Y_0) .

Step 2: random search according to smell. Fruit

flies define the random direction and pace in their flight according to the food odor concentration they feel, which can be calculated as

$$\begin{cases} X_i = X_0 + R, \\ Y_i = Y_0 + R, \end{cases} \quad (1)$$

where (X_i, Y_i) is the current location of the i th fruit fly and R denotes the random pace.

Step 3: calculation of the odor concentration decision value S_i . The food odor concentration decision value is the reciprocal of the distance to the origin of the coordinate axes for the i th fruit fly, which is calculated as

$$S_i = \frac{1}{\sqrt{X_i^2 + Y_i^2}}. \quad (2)$$

Step 4: calculation of the odor concentration value (fitness function value) Smell. The odor concentration value is obtained based on the odor-concentration decision value:

$$\text{Smell}_i = F(S_i), \quad (3)$$

where Smell_i refers to the food odor concentration value (fitness function value) felt by the i th fruit fly.

Step 5: finding the fruit fly that has the greatest value among Smell (optimal individual), described as follows:

$$\text{Smell}_{\text{best}} = \max \{ \text{Smell} \}, \quad (4)$$

where ‘best’ is the index of the optimal individual in the group whose odor concentration value is the largest.

Step 6: visual localization. Suppose that the current optimal individual is located in $(X_{\text{best}}, Y_{\text{best}})$ and its odor concentration value is $\text{Smell}_{\text{best}}$. Then the group flies toward the best individual. The process is presented as follows:

$$\begin{cases} \text{Smell}_i = \text{Smell}_{\text{best}}, \\ X_i = X_{\text{best}}, \\ Y_i = Y_{\text{best}}, \end{cases} \quad (5)$$

where $(X_{\text{best}}, Y_{\text{best}})$ indicates the position of the current optimal individual.

Step 7: iteration optimization. If the current iteration number $t < m$, repeat steps 2–5. Then determine if the current optimal odor concentration is better than the previous optimal value. If so, run step 6. The algorithm terminates when $t = m$.

3 Elimination-based fruit fly optimization algorithm

3.1 Reinforced vision search

It should be emphasized that in FOA, all fruit fly individuals will fly to the optimal individual directly, meaning that after step 6, all individuals in the group will arrive at the optimal individual. Thus, this process easily causes the algorithm to become trapped in local optima, resulting in a low optimization precision. Thus, in this study, we enhance the vision search ability of the fruit fly in the vision search process. In the process of flying to the food, other fruit flies constantly observe the location of the fruit fly that has the largest food odor concentration, and approach it gradually, but not by just rapidly and directly flying to the position of the best fruit fly. By adding the vision search ability, the convergence rate of the algorithm can be improved, while both the probability of falling into local optima and the running time are reduced. The vision search process can be described by

$$\begin{cases} X_i = c \cdot X_i + (1 - c) \cdot X_{\text{best}}, \\ Y_i = c \cdot Y_i + (1 - c) \cdot Y_{\text{best}}, \end{cases} \quad (6)$$

where c is a random number ranging from 0 to 1. Then calculate the odor concentration value Smell_i according to Eqs. (2) and (3).

3.2 Elimination mechanism

Since the foraging behavior of a fruit fly is simple, FOA can easily fall into local optima, resulting in a low optimization precision. In this study, we add an elimination mechanism to the foraging behavior of the fruit fly; i.e., some individuals in the group are eliminated and some new ones are generated in the fruit fly foraging process. This can not only increase the population diversity but also maintain the

convergence of the algorithm. Therefore, the scalability of the algorithm will be improved. The updating formula is

$$N = N - N(\text{worst}) \cup N(\text{new}), \quad (7)$$

where $N(\text{worst})$ represents the weak fruit flies in the group, $N(\text{new})$ denotes the newly generated fruit flies, and ‘-’ and ‘ \cup ’ refer to the set subtraction operation and set addition operation, respectively. The flow chart for EFOA is presented in Fig. 1.

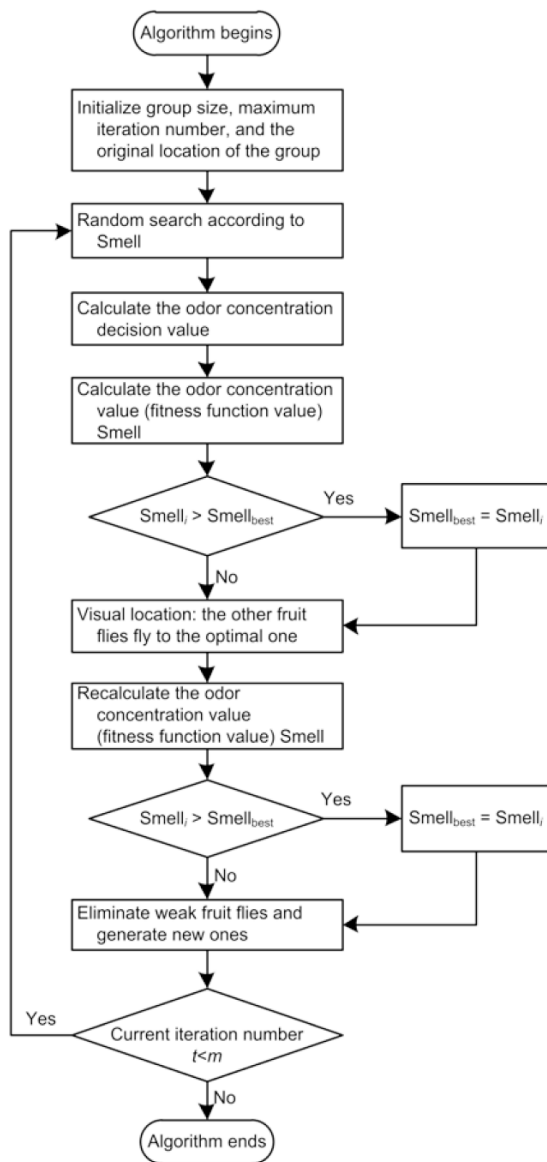


Fig. 1 Elimination-based fruit fly optimization algorithm working diagram

4 Elimination-based fruit fly optimization algorithm for solving the traveling salesman problem

In the process of using EFOA to solve TSP, an individual fruit fly represents a solution sequence. For a TSP of n cities, the solution sequence is a full permutation of integers from 1 to n . For instance, in a TSP of three cities, individual fruit fly (3, 1, 2) means an access path from city 3 to city 1 and then city 1 to city 2.

4.1 Definition of basic operators

Based on the proposed algorithm, two operators, a reverse operator and a multiplication operator, are defined. They are performed in the solution sequence in the smell and vision search processes, respectively.

Definition 1 (Reverse operator) The reverse operator is an evolution of the 2-opt algorithm (Croes, 1958). Suppose that the TSP solution sequence of n nodes is $F=(a_i)$ ($i=1, 2, \dots, n$), and its reverse term $TR(i, j)$ refers to nodes a_i and a_j in F ; then $F'=F \oplus TR(i, j)$ means reversing the subsequence between a_i and a_j to make them neighbors, leaving the rest of the solution sequence unchanged. The difference between a reverse operator and the 2-opt algorithm is in the selection of nodes a_i and a_j . In the reverse operator, node a_i is selected randomly, and a_j is the closest node to a_i in F .

For example, for a solution sequence $F(1, 2, 3, 4, 5)$, $i=2, j=5$, we have $F'=F \oplus TR(i, j)=(1, 2, 5, 4, 3)$.

Definition 2 (Multiplication operator) Given two solution sequences $A=(a_i)$ and $B=(b_i)$ ($i=1, 2, \dots, n$), we randomly select three adjacent nodes b_{i-1}, b_i , and b_{i+1} from B , while their corresponding nodes in A are a_j, a_k , and a_m , respectively. Then the multiplication operator will generate at least one of the following three subsequences through the reverse operator: (a_k, a_j) , (a_k, a_m) , and (a_j, a_k, a_m) .

Suppose that solution sequences A_1, A_2 , and A_3 are the results of three reverse operations on A . Then $A \otimes B$ returns the best one among them, which has the largest fitness value. The specific steps are

$$\begin{cases} A_1 = A \oplus TR(k, j), \\ A_2 = A \oplus TR(k, m), \\ A_3 = (A \oplus TR(k, j)) \oplus TR(k, m), \\ A \otimes B = \text{best}(A_1, A_2, A_3). \end{cases} \quad (8)$$

For example, given two solution sequences $A(2, 1, 6, 3, 7, 4, 5)$ and $B(1, 2, 3, 4, 5, 6, 7)$, if $b_i=3, b_{i-1}=2, b_{i+1}=4$, then $A_1=(6, 1, 2, 3, 7, 4, 5), A_2=(2, 1, 6, 3, 4, 7, 5)$, and $A_3=(6, 1, 2, 3, 4, 7, 5)$. The best one of them will be returned in $A \otimes B$.

4.2 Algorithm flow

Step 1: initialization. Define the group size as P , the maximum number of algorithm iterations as m , and the original location of the group as (X_0, Y_0) .

Step 2: random search according to smell. X_k^t represents the location of the k th fruit fly in the t th iteration, and an extra reverse operation is performed before the search. We should determine two nodes a_i and a_j in reverse commutator $TR(i, j)$ first:

$$\begin{cases} i = X_i^t(\text{rdx}), & 1 \leq \text{rdx} \leq n, \\ j = \text{the index of the node closest to } a_i \text{ in } X_i^t, \\ X_i^t = X_i^t \oplus TR(i, j), \end{cases} \quad (9)$$

where i is the node index in the solution sequence of the k th fruit fly in the t th iteration, rdx is a random integer ranging from 1 to n , j represents the location of the node that is closest to a_i in the solution sequence.

Step 3: calculating the odor concentration of X_i^t and comparing it with that of the current optimal individual. X_i^t is updated to be the new current optimal individual in the group if the odor concentration felt by X_i^t is better.

Step 4: visual localization. The other fruit flies fly to the optimal one through visual observation:

$$X_i^{t+1} = X_i^t \otimes X_{\text{best}}. \quad (10)$$

Step 5: repeating step 3. X_i^{t+1} is updated to be the new current optimal individual in the group if the odor concentration felt by X_i^{t+1} is better.

Step 6: elimination mechanism:

$$N = N - N(\text{worst}) \cup N(\text{new}). \quad (11)$$

Eq. (11) is similar to Eq. (7) except that $N(\text{worst})$ has only 10% of the fruit flies in the group.

Step 7: iteration optimization. If the current iteration number $t < m$, repeat steps 2–6. The algorithm terminates when $t = m$.

5 Experiment results

Ten benchmarks are selected from TSPLIB to test the performance of EFOA. Every benchmark is tested 20 times.

Table 1 shows the results of the experiments repeated 20 times. BKS refers to the theoretical value of the dataset, Average represents the average value of the experimental results, SD is the standard deviation in the experiments, and Error is defined as

$$\text{Error} = \frac{\text{Average} - \text{BKS}}{\text{BKS}} \times 100\%. \quad (12)$$

Figs. 2a–2d present the results of the iteration optimization using the basic FOA and the improved version (EFOA) on the Berlin52, Eil51, Lin105, and

Table 1 Results from the elimination-based fruit fly optimization algorithm to solve traveling salesman problem

Dataset	BKS	Best	Worst	Average	SD	Error (%)
Berlin52	7542	7542	7542	7542.00	0	0
Eil51	426	426	431	427.53	1.2600	0.359
Eil76	538	540	550	544.05	2.9100	1.125
St70	675	675	682	677.26	2.3300	0.335
Rat99	1211	1217	1232	1237.20	15.1700	2.164
Kroa100	21 282	21 282	21 527	21 357.00	43.7700	0.352
Krob100	22 141	22 219	22 419	22 355.00	65.7300	0.967
Eil101	629	635	649	642.05	4.7700	2.075
Lin105	14 379	14 379	14 553	14 427.06	44.6700	0.334
Ch150	6528	6558	6660	6618.20	31.6837	1.382

Every benchmark was tested 20 times. BKS: theoretical value of the dataset; SD: standard deviation

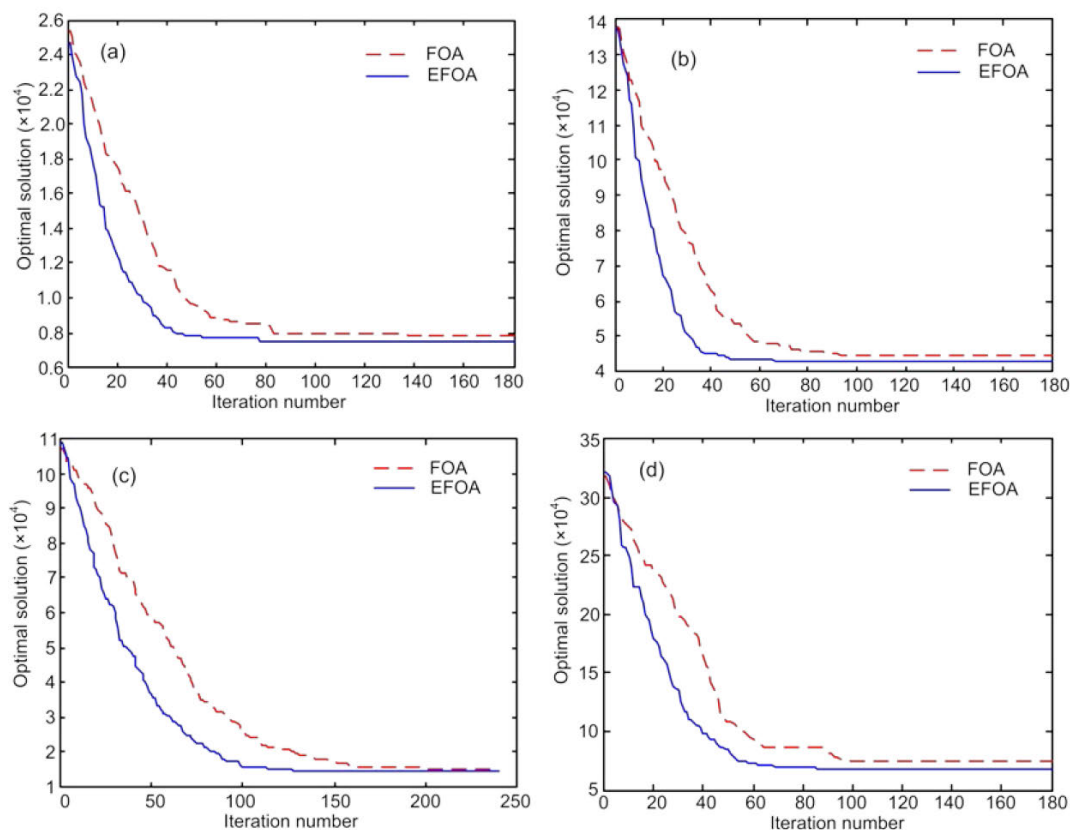


Fig. 2 Comparison of iteration optimization using fruit fly optimization algorithm (FOA) and elimination-based FOA (EFOA) on the Berlin52 dataset with 7542-opt (a), the Eil51 dataset with 426-opt (b), the Lin105 dataset with 14379-opt (c), and the St70 dataset with 675-opt (d)

St70 datasets, respectively. The solid line represents EFOA, while the dashed line represents FOA. It can be seen that, in the improved version (EFOA), the optimization result does not change after 40 iterations, while the values for the basic version (FOA) continue changing after even more than 100 iterations. In the end, the solid line is beneath the dashed line. This shows that EFOA can find the optimal solution after just 40 iterations while FOA needs more than 100 iterations. The former has a much higher optimization precision than the latter when solving TSP. Since EFOA reinforces vision search in the foraging behavior of the fruit flies and adds an elimination mechanism to FOA, it consequently improves the convergence rate and increases the diversity of the group. Thus, it avoids falling into local optima and finally improves the optimization precision.

Fig. 3 presents the theoretical and actual values obtained using the improved FOA (EFOA). We can see that the theoretical values and actual values have little difference, and the theoretical values are reached

when using the improved FOA on the Berlin52, Eil51, St70, Kroa100, and Lin105 datasets.

Table 2 shows the results obtained using EFOA and the alternatives to solve TSP. The optimal value of each dataset is marked in bold. It can be seen that EFOA achieves results of 7542.00, 427.53, 677.26, 1237.20, and 642.05, which are the best compared with the alternatives on the Berlin52, Eil51, St70, Rat99, Eil101, and Ch150 datasets.

However, for the Eil76, Krob100, and Kroa100 datasets, the best performance is 542.00 with CGAS (Dong *et al.*, 2012), 22 336.20 with DWIO (Zhou *et al.*, 2015), and 21 289.98 with DWIO, while the proposed algorithm achieves results of 544.05, 22 355.00, and 21 357.00, which are comparable with the best performances. As for the Lin105 dataset, the four algorithms achieve a similar performance. Each dataset has its own characteristics; thus, it can be difficult for a specific algorithm to achieve the optimal value in all of the datasets. From the comparisons above, we can see that EFOA improves the

Table 2 Comparison among the elimination-based fruit fly optimization algorithm (EFOA) and other alternatives for solving the traveling salesman problem

Algorithm	Theoretical value														
	Berlin52			Eil51			Eil76			St70			Rat99		
	Average	SD	Error (%)	Average	SD	Error (%)	Average	SD	Error (%)	Average	SD	Error (%)	Average	SD	Error (%)
oRABNET (Pasti and de Castro, 2006)	8073.97	270.14	7.05	438.70	3.52	2.98	556.10	8.03	3.36	-	-	-	-	-	-
RABNET (Masutti and de Castro, 2009)	7932.50	277.25	5.18	437.47	4.20	2.69	556.33	5.30	3.41	-	-	-	-	-	-
HACO (Wu and Ouyang, 2012)	7560.54	67.48	0.23	431.20	2.00	1.22	-	-	-	-	-	-	1241.33	9.60	1.42
CGAS (Dong et al., 2012)	7634.00	-	1.22	-	-	-	542.00	-	0.74	-	-	-	-	-	-
ACOTM (Peker et al., 2013)	7635.40	-	1.24	435.40	-	2.21	656.50	-	5.11	-	-	-	-	-	-
HA (Gündüz et al., 2015)	7544.37	0	0.03	443.39	5.25	4.08	557.98	4.10	3.71	700.58	7.51	3.79	-	-	-
DWIO (Zhou et al., 2015)	7544.36	0	0.03	428.98	0.35	7.00	-	-	-	677.30	0.34	0.35	-	-	-
EFOA	7542.00	0	0	427.53	1.26	0.36	544.05	2.91	1.15	677.26	2.33	0.34	1237.20	15.17	2.16
Algorithm	Theoretical value														
	Kroa100			Krob100			Eil101			Lin105			Ch150		
	Average	SD	Error (%)	Average	SD	Error (%)	Average	SD	Error (%)	Average	SD	Error (%)	Average	SD	Error (%)
oRABNET (Pasti and de Castro, 2006)	21 868.47	245.76	2.76	-	-	-	654.83	6.57	4.11	14 702.2	328.0	2.25	6753.20	83.01	3.45
RABNET (Masutti and de Castro, 2009)	21 522.73	93.34	1.13	-	-	-	648.63	3.85	3.12	14 400.7	44.0	0.15	6738.37	76.14	3.22
HACO (Wu and Ouyang, 2012)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CGAS (Dong et al., 2012)	21 437.00	-	0.73	-	-	-	-	-	-	-	-	-	-	-	-
ACOTM (Peker et al., 2013)	21 567.10	-	1.34	-	-	-	655.00	-	4.13	14 475.2	-	0.67	-	-	-
HA (Gündüz et al., 2015)	22 435.31	231.34	5.42	-	-	-	683.39	6.56	8.65	-	-	-	6677.12	19.30	2.28
DWIO (Zhou et al., 2015)	21 289.98	0.02	0.04	22 336	0.19	0.88	-	-	-	-	-	-	-	-	-
EFOA	21 357.00	43.77	0.35	22 355	65.73	0.97	642.05	4.77	2.08	14 427.1	44.7	0.33	6618.20	31.68	1.38

Every benchmark was tested 20 times. The optimization value of each dataset is marked in bold. BKS: theoretical value of the dataset; SD: standard deviation

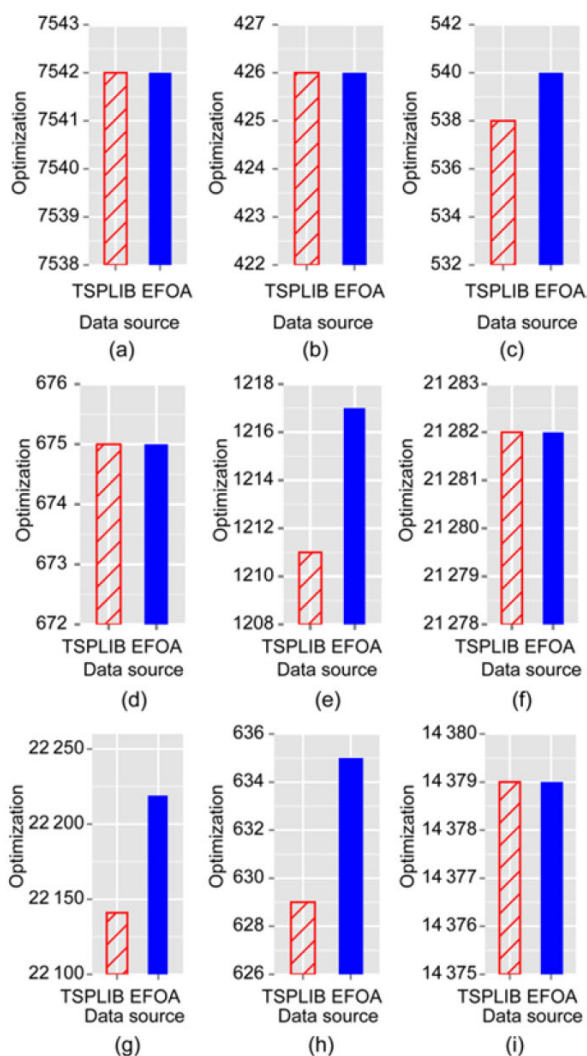


Fig. 3 Comparison between theoretical and actual values obtained with EFOA on the Berlin52 (a), Eil51 (b), Eil76 (c), St70 (d), Rat99 (e), Kroa100 (f), Krob100 (g), Eil101 (h), and Lin105 (i) datasets

optimization precision in most datasets, showing the effectiveness of the proposed algorithm.

6 Conclusions

In this paper, an elimination-based fruit fly optimization algorithm (EFOA) was proposed. In EFOA, the vision search ability of fruit flies is reinforced in their foraging behaviors to avoid trapping the algorithm in local optima, and an elimination mechanism is added into FOA. The proposed method improves the convergence rate and the optimization

precision of the result. Two operators, a reverse operator and a multiplication operator, were proposed when using the improved FOA to solve TSP. Ten datasets in TSPLIB were tested to validate our method. The comparison results demonstrate that our method has much better optimization precision and stability than other methods.

However, large-scale datasets for TSP were not tested in this study. Our future research investigations include exploring how to use this algorithm to obtain a more satisfactory solution for large-scale TSPs in shorter time.

References

- Bellman, R.E., Dreyfus, S.E., 1962. Applied Dynamic Programming. Princeton University Press, New Jersey, USA, p.50-68.
- Clerc, M., 2004. Discrete particle swarm optimization, illustrated by the traveling salesman problem. *In: Onwubolu, G.C., Babu, B.V. (Eds.), New Optimization Techniques in Engineering*. Springer Berlin Heidelberg, p.219-239. https://doi.org/10.1007/978-3-540-39930-8_8
- Croes, G.A., 1958. A method for solving traveling-salesman problems. *Oper. Res.*, **6**(6):791-812. <https://doi.org/10.1287/opre.6.6.791>
- Ding, C., Cheng, Y., He, M., 2007. Two-level genetic algorithm for clustered traveling salesman problem with application in large-scale TSPs. *Tsinghua Sci. Technol.*, **12**(4):459-465. [https://doi.org/10.1016/S1007-0214\(07\)70068-8](https://doi.org/10.1016/S1007-0214(07)70068-8)
- Dong, G.F., Guo, W.W., Tickle, K., 2012. Solving the traveling salesman problem using cooperative genetic ant systems. *Exp. Syst. Appl.*, **39**(5):5006-5011. <https://doi.org/10.1016/j.eswa.2011.10.012>
- Dorigo, M., Gambardella, L.M., 1997. Ant colonies for the travelling salesman problem. *BioSystems*, **43**(2):73-81. [https://doi.org/10.1016/S0303-2647\(97\)01708-5](https://doi.org/10.1016/S0303-2647(97)01708-5)
- Escario, J.B., Jimenez, J.F., Giron-Sierra, J.M., 2015. Ant colony extended: experiments on the travelling salesman problem. *Exp. Syst. Appl.*, **42**(1):390-410. <https://doi.org/10.1016/j.eswa.2014.07.054>
- Geng, X.T., Chen, Z.H., Yang, W., et al., 2011. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. *Appl. Soft Comput.*, **11**(4):3680-3689. <https://doi.org/10.1016/j.asoc.2011.01.039>
- Grefenstette, J.J., Gopal, R., Rosmaita, B.J., et al., 1985. Genetic algorithms for the traveling salesman problem. 1st Int. Conf. on Genetic Algorithms and Their Applications, p.160-168.
- Gündüz, M., Kiran, M.S., Özceylan, E., 2015. A hierarchic approach based on swarm intelligence to solve the traveling salesman problem. *Turk. J. Electric. Eng. Comput. Sci.*, **23**(1):103-117. <https://doi.org/10.3906/elk-1210-147>

- Hendtlass, T., 2003. Preserving diversity in particle swarm optimisation. In: Chung, P.W.H., Hinde, C., Ali, M. (Eds.), *Developments in Applied Artificial Intelligence*. Springer Berlin Heidelberg, p.31-40.
https://doi.org/10.1007/3-540-45034-3_4
- Hoffmann, M., Mühlenthaler, M., Helwig, S., et al., 2011. Discrete particle swarm optimization for TSP: theoretical results and experimental evaluations. In: Bouchachia, A. (Ed.), *Adaptive and Intelligent Systems*. Springer Berlin Heidelberg, p.416-427.
https://doi.org/10.1007/978-3-642-23857-4_40
- Jolai, F., Ghanbari, A., 2010. Integrating data transformation techniques with Hopfield neural networks for solving travelling salesman problem. *Exp. Syst. Appl.*, **37**(7): 5331-5335. <https://doi.org/10.1016/j.eswa.2010.01.002>
- Karaboga, D., Gorkemli, B., 2011. A combinatorial artificial bee colony algorithm for traveling salesman problem. *IEEE Int. Symp. on Innovations in Intelligent Systems and Applications*, p.50-53.
<https://doi.org/10.1109/INISTA.2011.5946125>
- Kirkpatrick, S., 1984. Optimization by simulated annealing: quantitative studies. *J. Stat. Phys.*, **34**(5-6):975-986.
<https://doi.org/10.1007/BF01009452>
- Lawler, E.L., Wood, D.E., 1966. Branch-and-bound methods: a survey. *Oper. Res.*, **14**(4):699-719.
<https://doi.org/10.1287/opre.14.4.699>
- Little, J.D.C., Murty, K.G., Sweeney, D.W., et al., 1963. An algorithm for the traveling salesman problem. *Oper. Res.*, **11**(6):972-989. <https://doi.org/10.1287/opre.11.6.972>
- Liu, F., Zeng, G.Z., 2009. Study of genetic algorithm with reinforcement learning to solve the TSP. *Exp. Syst. Appl.*, **36**(3):6995-7001.
<https://doi.org/10.1016/j.eswa.2008.08.026>
- Mahi, M., Baykan, Ö.K., Kodaz, H., 2015. 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.
<https://doi.org/10.1016/j.asoc.2015.01.068>
- Masutti, T.A.S., de Castro, L.N., 2009. A self-organizing neural network using ideas from the immune system to solve the traveling salesman problem. *Inform. Sci.*, **179**(10):1454-1468. <https://doi.org/10.1016/j.ins.2008.12.016>
- Ouyang, X.X., Zhou, Y.G., Luo, Q.F., et al., 2013. A novel discrete cuckoo search algorithm for spherical traveling salesman problem. *Appl. Math. Inform. Sci.*, **7**(2): 777-784. <https://doi.org/10.12785/amis/070248>
- Pan, W.T., 2011. *Fruit Fly Optimization Algorithm*. Tsang Hai Book Publishing Co., Taipei, China, p.221-232 (in Chinese).
- Pan, W.T., 2012. A new fruit fly optimization algorithm: taking the financial distress model as an example. *Knowl.-Based Syst.*, **26**:69-74.
<https://doi.org/10.1016/j.knsys.2011.07.001>
- Pasti, R., de Castro, L.N., 2006. A neuro-immune network for solving the traveling salesman problem. *IEEE Int. Joint Conf. on Neural Network*, p.3760-3766.
<https://doi.org/10.1109/IJCNN.2006.247394>
- Peker, M., Şen, B., Kumru, P.Y., 2013. An efficient solving of the traveling salesman problem: the ant colony system having parameters optimized by the Taguchi method. *Turk. J. Electric. Eng. Comput. Sci.*, **21**(55):2015-2036.
<https://doi.org/10.3906/elk-1109-44>
- Wu, J.Q., Ouyang, A.J., 2012. A hybrid algorithm of ACO and delete-cross method for TSP. *IEEE Int. Conf. on Industrial Control and Electronics Engineering*, p.1694-1696.
<https://doi.org/10.1109/ICICEE.2012.448>
- Zhou, Y.Q., Luo, Q.F., Chen, H., et al., 2015. A discrete invasive weed optimization algorithm for solving traveling salesman problem. *Neurocomputing*, **151**:1227-1236.
<https://doi.org/10.1016/j.neucom.2014.01.078>