# HMFA: A Hybrid Mutation-Base Firefly Algorithm for Travelling Salesman Problem

**Mohammad Saraei and Parvaneh Mansouri**

**Abstract** The Travelling Salesman Problem (TSP) is one of the major problems in graph theory and also is NP-Hard Problem. In this work, by improving the firefly algorithm (MFA), we introduced a new method for solving TSP. The result of the proposed method has compared with the other algorithms such as Firefly algorithm, GA and PSO. The Proposed Method out performs of other algorithms.

**Keywords** NP-Hard problem · Travelling salesman problem (TSP) Firefly algorithm (FA) · Swapping mutation · Inversion mutation Insertion mutation · 2-Opt mutation

## 1 Introduction

Traveling salesman problem (TSP) is one of the known problems in artificial intelligence. TSP is a discrete optimization problem and also NP-Hard problem. Many Studies have been done to find the best solution for this problem, but no exact solution has been provided yet. This simple problem has many applications, including vehicle timing [1] route optimization to transport the goods to different locations, route optimization for postal shipments, Vehicle routing [2] and route minimizing of a tour. The TSP represents the salesman who wants to visit a set of cities exactly once and finally turns back to the starting city. The objective is to determine the tour with minimum total distance (see Fig. 1).

M. Saraei (✉) · P. Mansouri
Department of Computer, Faculty of Technical, Islamic Azad University, Arak Branch, Arak, Iran
e-mail: mohammadsaraei@gmail.com

M. Saraei
Young Researchers and Elite Club, Islamic Azad University, Arak Branch, Arak, Iran

P. Mansouri
Department of Computer Science, Delhi University, Delhi, India

S. Montaser Kouhsari (ed.), *Fundamental Research in Electrical Engineering*, Lecture Notes in Electrical Engineering 480,
https://doi.org/10.1007/978-981-10-8672-4_31

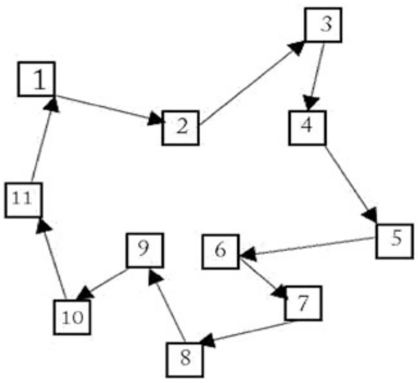


**Fig. 1** Sample of solving TSP

In recent years many approaches have been developed to solve TSP The simplest exact method solve all possible tours, and then select the optimal tour with the minimum total cost. All possible permutations of N cities are equal to $N!$, so, every tour can be represented in 2N different manner depends on the initial city and the length of tour. So the size of search space is computed as Eq. (1). It is obviously that this measurement is not possible for computational time even for 50 cities.

$$T = \frac{N!}{2 * N} = \frac{(N-1)!}{2} \tag{1}$$

In Sect. 2, we examine some algorithms provided for solving the TSP. In addition, a mathematical model and an introduction to Firefly algorithm will be described. In Sect. 3, we consider the proposed algorithm. We indicate the desired mutations used in the proposed algorithm. Then, in Sect. 4, the proposed algorithm will be used to solve the TSP and the results will be compared with other bio-inspired algorithms such as GA, PSO and FA. In Sect. 5, we review the strengths of the proposed method compared to other algorithms. Finally the paper ends with Conclusion and Acknowledgement.

## 2 Theoretical Principles

Recently, different methods have been proposed to solve the TSP whichever have their own strengths and weaknesses but it is important to use the method or algorithm that achieves the best tour in the shortest possible time. Some meta-heuristic algorithms used to solve the TSP are: Genetic algorithm (GA) [3–5]. Particle swarm (PSO) [6–8], Ant colony (ACO) [9–11], Memetic algorithms [12], Artificial Bee Colony [13, 14], Bee Colony [15, 16], and etc. Better solution can be achieved by changing the parameters in any of these algorithms. In 2014, Saranya et al. have presented a method for solving the TSP based on Tabu search and biological algorithms such as ant colony optimization algorithm, cuckoo algorithm and bee algorithm [17]. In 2012, Yang et al. have purposed an optimization approach to reduce the processing costs associated with ant colony routing (ACO) and they have Improved ACO using individual diversification strategy [18]. So that, the speed of ACO greatly increased. They used this approach for solving the TSP. Rizak Allah et al. In 2013 have presented a hybrid algorithm called ACO—FA, that integrate Ant colony algorithm (ACO) and Firefly algorithm (FA) to solve unlimited problems [19]. Las zolocota [20] used Firefly algorithm to solve multiple TSP. In this work, we purpose an accurate and fast algorithm to solved TSP by adding best and Effective mutations to FA.

## 2.1 Mathematical Model of Travelling Salesman Problem

In this study, our purpose is to find best tour of symmetric TSP. In symmetric TSP, the distance from city A to city B equals to distance from city B to city A. However, in asymmetric mode, the distance from city A to city B is not necessarily equal to the distance of city A to city B. We can consider the symmetric TSP problem as a complete and undirected graph where

$$A = \{(i,j) : i,j \in V, i \neq j\}$$

'A' is a set of edges and $V = \{0, \ldots, N\}$ is a set of nodes.

The number of possible solutions are $\frac{1}{2}(\mathrm{N} - 1)$, (N is the number of cities and $\mathrm{N} > 2$). In fact, the number of possible solutions are equal to the number of Hamiltonian cycles in a complete graph with N nodes. The mathematical form of the objective cost function of TSP is as follows:

$$\min \mathrm{z} = \sum_{\mathrm{i}=0}^{\mathrm{N}} \sum_{\mathrm{j} \neq \mathrm{i}, \mathrm{j}=0}^{\mathrm{N}} \mathrm{c}_{\mathrm{ij}} \tag{2}$$

where $c_{\mathrm{ij}}$ is the distance between nodes $i$ and $j$. $i$, $j = 0,1, \ldots, \mathrm{N}$

## 2.2 Distance of Two Cities

For computing the distance between two cities (nodes), there are some methods such as hamming and Euclidean distance formulas. We consider the cities as nodes of two-dimensional Cartesian space (x, y) and by using Euclidean distance formula as follows:

$$d(i,j) = d(j,i) \tag{3}$$

$$d_{ij} = \sqrt{\left(x_i - x_j\right)^2 - \left(y_i - y_j\right)^2} \tag{4}$$

## 3 Firefly Algorithm

In 2009 Yang [21] introduced the optimization firefly algorithm (FA). FA has inspired by fireflies that use short and rhythmic lights to attract the hunt, protection or attract mates systems. There are two important issues in firefly algorithm: changes in light intensity and formulating the attraction. For simplicity, we can always assume that its light determines the attraction of firefly, which in turn is associated with the objective function. The attraction is proportional to brightness and a firefly with lower light is absorbed to firefly with brighter light, and if there is no light, it moves randomly. The firefly will be visible only for a limited period due to distance and light reduction by air. A firefly can be considered as a point light source.

We know that the light intensity at a certain distance r from the light source follows the inverse square law. The law states that the light intensity I decrease by increasing the distance r.

$$I \propto \frac{1}{r} \tag{5}$$

As mentioned, by increasing distance of two fireflies, the light intensity of between them is going to be weaker and weaker. In the simplest case, we can consider the light intensity of a point source by analysis factor γ, in distance r as Eq. (5) ($I_0$ is the light intensity in r = 0).

Since the attraction of firefly is proportional to light intensity seen by adjacent firefly, the attraction of fireflies is defined as Eq. (6) ($\beta_0$ is the attraction in r = 0).

$$\beta(r) = \beta_0 e^{-\gamma r^2} \tag{6}$$

The distance between any two fireflies $i$ and j at $x_i$ and $x_j$, respectively, is the Cartesian distance:

$$r_{ij} = \sqrt{\sum_{k=1}^{d} \left(x_{i,k} - x_{j,k}\right)^2} \tag{7}$$

where $x_{i,k}$ is the $k$ th component of the spatial coordinate $x_i$ of $i$ th firefly.

Brightness is also proportional to objective function. Therefore, updating the location for each pair of fireflies $i$ and $j$ at $x_i$ and $x_j$ is as following equation:

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}\left(x_j^t - x_i^t\right) + \alpha_t \varepsilon_t \tag{8}$$

The firefly algorithm has been formulated by following properties:

1. All fireflies are single type, so that a firefly attracts all other fireflies.
2. Attraction is proportional to brightness and a firefly with lower light is absorbed to firefly with brighter light.
3. If there is no firefly brighter than the other firefly, then the firefly moves randomly.

The pseudo code of FA as follows:

**Algorithm 1: The pseudo code of Firefly algorithm**

```
Firefly Algorithm:
Objective function f(x),x = $(x_1,\ldots,x_d)^T$
Generate initial population of firefly $x_i$ (i = 1,2,…,n)
Light intensity $I_i$ at $x_i$ is determined by f($x_i$)
Define light absorption coefficienty
While (t < MaxGeneration)
for i = 1: n all n fireflies
for j = 1: n all n fireflies
if ($I_j > I_i$),Move firefly I towards j in d _dimension;end if
Evaluate new solutions and update light intensity
end for j
end for i
Rank the fireflies and find the current best
End while
Postprocess result and visualization Rank the fireflies and find the current best;
End while;
Post process results and visualization;
End procedure
```

## 3.1 Some Mutation Operators

In FA, for finding the shortest path, we can use the various mutations such as: random, inversion, swapping and greedy mutations and, etc.

This will prevent the falling into the trap local optimal. With this work new solution will be replacement of previous solutions.

### 3.1.1 Swapping Mutation

This method is the most commonly used methods; this mutation can be performed on a couple of points. The operation of this mutation for two points, swap the two points randomly shown in the Fig. 2.

### 3.1.2 Inversion Mutation

In this method two elements are randomly selected and then we inverse enclosed elements in block between them and place them on their own place (see Fig. 3).

### 3.1.3 Insertion Mutation

In this method two elements are randomly selected and transfer one of the two elements after other chosen element (see Fig. 4).

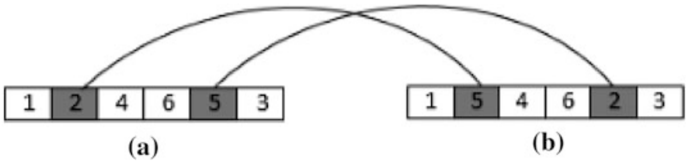


**Fig. 2** Swapping mutation. **a** Before mutation, **b** after mutation

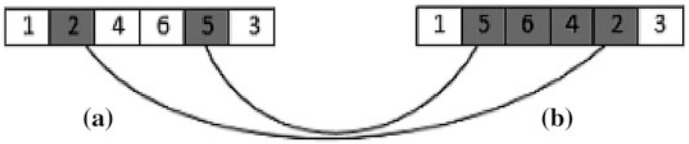
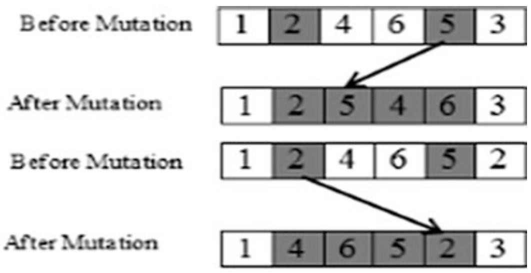


**Fig. 3** Inversion Mutation. **a** Before mutation, **b** after mutation

**Fig. 4** Insertion mutation



### 3.1.4 2-Opt Mutation

This mutation is one of the most effective mutations for improvement in finding optimal tour in TSP.In this mutation via check path and selection four points with k distance between them, if the way of their connection contains twist, will open twist path and decreases the path length. Figure 4a optimized route is displayed in Fig. 4b.

The process of execution of this mutation is as follows:

We select randomly four points I, j, n, m

1. We calculate the distance between the points using following relation.
2. |(i, j)| + |(n, m)| > |(i, m)| + |(n, j)|.
3. If the relation is true we change the value of element j with the value of element m.
4. If necessary we repeat this process and select other points.

Changes resulting from this mutation on the tour are as follows (Figs. 5 and 6):

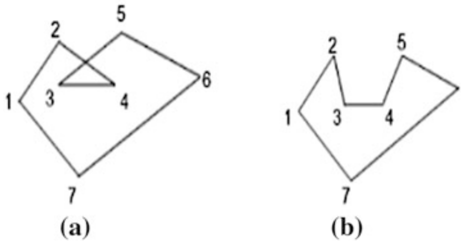


**Fig. 5** **a** First tour (before mutation), **b** second tour (after mutation)

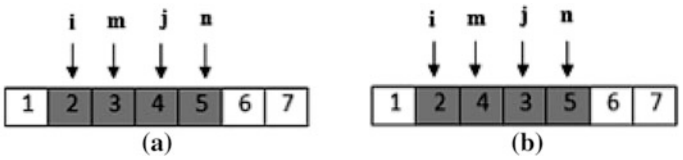


**Fig. 6** **a** solution before mutation, **b** solution after mutation

## 4 Proposed Method

Sometimes, in practice it is possible the FA to be trapped in a local minimum and the rapid convergence there. This is not to achieve an appropriate response. Therefore, to solve the TSP our algorithm should have the least amount of complexity, because complexity increases the running time of algorithm. In this study, for escaping of the local minimum at acceptable running time of algorithm, our purpose is to add some mutations in FA to find a faster algorithm without trapped in a local minimum. An operator develops a mutation that causes widening areas are discovered. In addition to the TSP graph, edges should not be crossed. Because the cross is to increase the length of the tour. Using the appropriate responses to increase the mutation rate. To fix the problem of trapped in a local minimum, instead of using an operator or a combination of some operators; we add randomly one of operators in each iteration of FA. Each mutation operation of Sect. 3.1.4 can able to solve just some special difficulties to determine shortest paths. In Table 1, we can see the results of solving salesman problem with four cities, population size are 10 and the number of iterations is 700. The structure of the proposed algorithm is as follows (Fig. 7):

**Step1**:
Create Model Of Benchmark TSP Problem.
**Step2**:
Objective Function F (Tour), Tour is a Matrix Contains Number of Cities.
**Step3**:
Define Parameters Algorithm Such as Max Iteration, number of population,delta, gamma,alpha, …
**Step4**:
Generate Initial Population Randomly {each Member of the Population is a Tour}.
**Step5**:
Calculate $r_{i,j}$,$r_{j,I}$ is equal: norm {tour(i)-tour(j)}.
**Step6**:
Evaluated New Tour and Calculate Cost.
**Step7**:
Create Random Number Between [1 to 4]
**Step8**:
**Switch** on Random Number Obtained in Above stage.
**Step9**:
Evaluate New Tour Obtained Of Mutation and Update Cost.
**Step10**:
**If (**Cost of (newer tour) < cost of (new tour)).
**Then** The newer tour is replaced tour **Else** The new tour is replaced tour.

In MFA (Combine four Mutation) in each iteration, randomly one of the mutation operators (swapping, inversion, Insertion, 2-opt) added to FA and TSP

**Table 1** Comparison between mutations for TSP with 700 iteration and 10 populations

| TSP problem | Swapping | | Insertion | | Reversion | | 2-Opt | | Combine | |
|---|---|---|---|---|---|---|---|---|---|---|
| Name | Best | Average | Best | Average | Best | Average | Best | Average | Best | Average |
| Ulysses16 | 7589.0436 | 7909.055 | 7521.3761 | 8295.5414 | 7399.7618 | 7413.972 | 7461.4804 | 7504.5579 | 7398.7618 | 7404.0949 |
| Ulysses22 | 8402.0016 | 8776.9591 | 8418.4765 | 11,372.2573 | 7550.8818 | 7609.0946 | 7799.3926 | 8402.8132 | 7530.9701 | 7573.5835 |
| Gr24 | 1549.3962 | 1638.6441 | 1397.3769 | 1691.6728 | 1288.6966 | 1298.6519 | 1421.0923 | 1505.8183 | 1279.5031 | 1317.1837 |
| Eil51 | 1127.8884 | 1127.8884 | 1103.0047 | 1103.0047 | 556.4436 | 562.3819 | 545.9919 | 575.1371 | 430.8606 | 445.1045 |

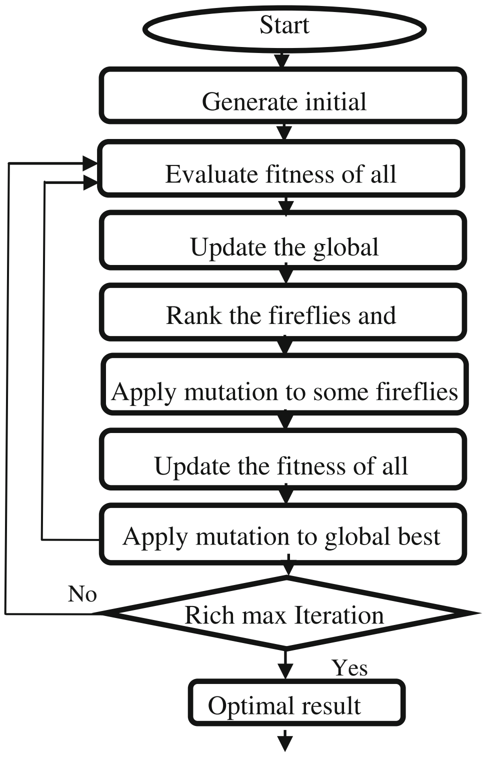


**Fig. 7** The flowchart of the proposed algorithm

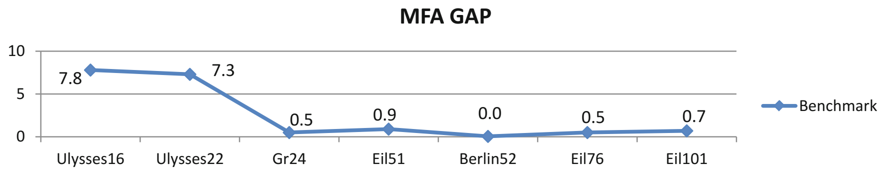


**Fig. 8** The GAP Diagram of the MFA algorithm to benchmarks

cost function tour is computed. Then, combined mutation (by using the method mentioned earlier) apply to global best solution. After some iteration minimum value of cost function will be achieve. Comparison of proposed algorithm and other algorithms such as FA, GA and PSO show that the proposed algorithm outperforms of others. The results show that the random combination of four mutations of Sect. 3.1.4, gives us the better solution.

## 5 Simulation

In this paper, using the improved FA to solve the standard TSP by MATLAB 2013 on a platform with Intel CORE i5, 2 GB RAM, and Windows 7 operating system's has been solved for standard algorithms such as GA, PSO and also FA. The average of 10 times running for each standard library problems TSPLIB [22] was calculated and the results have been compared with the results of the proposed algorithm in this paper. Also in Table 2 the benchmarks are used is visible. Respectively. Setting parameter GA and PSO and FA can be seen in Tables 3, 4 and 5. Table 6 shows the

**Table 2** Benchmark of TSP

| Benchmark problem | N.City | Optimal |
|---|---|---|
| Ulysses16 | 16 | 6859 |
| Ulysses22 | 22 | 7013 |
| Gr24 | 24 | 1272 |
| Eil51 | 51 | 426 |
| Berlin52 | 52 | 7542 |
| Eil76 | 76 | 538 |
| Eil101 | 101 | 629 |

**Table 3** Defined parameters for firefly

| Parameters | Value |
|---|---|
| Maximum number of iterations | 700 |
| Number of fireflies | 10 |
| Light absorption coefficient | 2 |
| Attraction coefficient base value | 1 |
| Mutation coefficient | 0.2 |

**Table 4** Defined Parameters for PSO

| Parameters | Value |
|---|---|
| Maximum number of iterations | 700 |
| Population size (Swarm size) | 10 |
| Inertia weight | 1 |
| Inertia weight damping ratio | 0.99 |
| Personal learning coefficient | 0.2 |
| Global learning coefficient | 0.4 |

**Table 5** Defined Parameters for GA

| Parameters | Value |
|---|---|
| Maximum number of iterations | 700 |
| Population size | 10 |
| Crossover percentage | 0.5 |
| Mutation percentage | 0.5 |

**Table 6** Comparison between FA, GA, PSO, and MFA for TSP with 1000 iteration and 10 populations

| TSP problem | GA | | Besttime (s) | PSO | | Best time (s) | FA | | Best time (s) | MFA | | Best time (s) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | Best | Average | | Best | Average | | Best | Average | | Best | Average | |
| Ulysses16 | 7539.5764 | 8270.5777 | 20.837 | 8342.7014 | 9683.9225 | 15.8817 | 9218.9743 | 10,030.5512 | 25.0046 | 7398.7618 | 7404.0949 | 8.0857 |
| Ulysses22 | 9019.0812 | 9610.0365 | 23.9897 | 9227.943 | 11,503.3717 | 15.6765 | 10,772.1213 | 12,182.9003 | 26.1966 | 7530.9701 | 7573.5835 | 11.4056 |
| Gr24 | 1657.9434 | 1703.6828 | 20.65 | 2073.9016 | 2401.5683 | 17.0012 | 2224.2237 | 2485.3758 | 23.7375 | 1279.5031 | 1317.1837 | 12.0406 |
| Eil51 | 702.1431 | 757.4782 | 22.6451 | 1113.7946 | 1223.0594 | 17.5213 | 1094.3539 | 1199.7793 | 26.7765 | 430.8606 | 445.1045 | 34.8505 |
| Berlin52 | 12,831.6181 | 14,141.9204 | 23.172 | 19,279.1045 | 22,368.3565 | 17.9856 | 19,578.6951 | 21,788.4744 | 29.693 | 7546.8573 | 7714.4453 | 51.4121 |
| Eil76 | 1105.1721 | 1199.2329 | 24.7965 | 1800.4436 | 1991.7083 | 18.4846 | 1564.0598 | 1791.851 | 28.079 | 545.7331 | 568.7338 | 73.1231 |
| Eil101 | 1650.7169 | 1754.5091 | 26.1545 | 2639.2026 | 2909.4256 | 17.8548 | 2385.9235 | 2515.0022 | 27.9111 | 634.8189 | 668.6202 | 88.0942 |

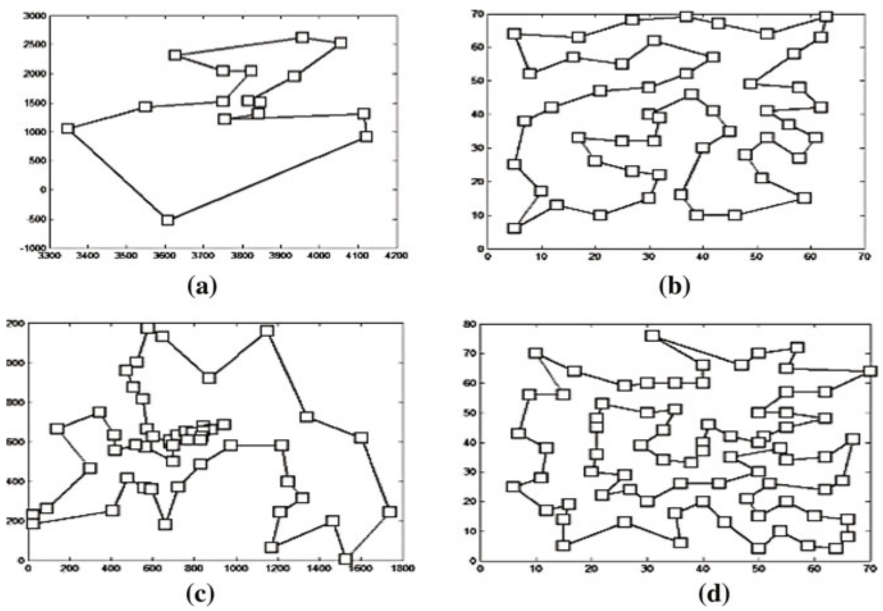
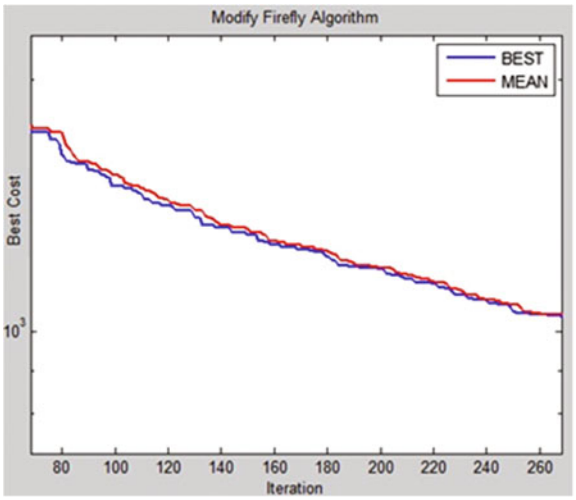


**Fig. 9** **a** The simulation result for gr24. **b** The simulation result for Eil51. **c** The simulation result for Berlin52. **d** The simulation result for Eil76

**Fig. 10** The output graph of MFA algorithm for Eil76



results for size population 10 and the number of iteration: 1000 in applied problems. The simulation results have been presented as graphical output in Fig. 9. And cost and roc curve Diagram for Eil76 (TSP Problem) displayed in Figs. 10, 11 in addition Fig. 8 shows the gap values of the MFA algorithm for Benchmarks, where

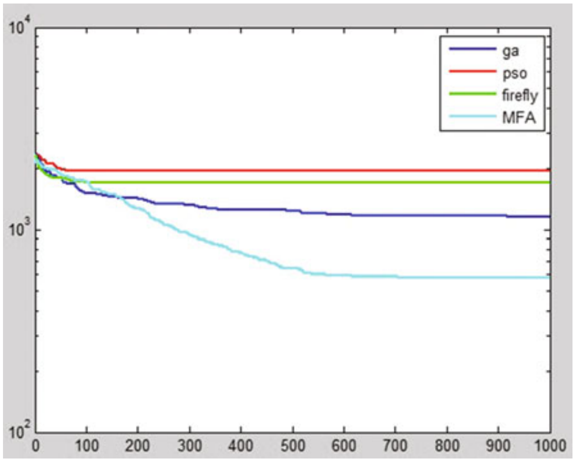


**Fig. 11** The rocurve graph of algorithms for Eil76

the gap is defined as the percentage of deviation. In this formula (A′) best answer of found by our algorithm and (A) the best answer Known (Optimal) for Benchmarks. The gap is calculated as follows:

$$GAP = \frac{C(A') - C(A)}{C(A)} \times 100 \tag{9}$$

## 6 Discussion and Conclusion

In this paper, a novel meta-heuristic algorithm called improved FA was applied to solve the standard TSP and we compared the performance of the three famous SI (swarm intelligence) algorithms include of GA, PSO, FA to solve TSP. The FA algorithm has a strong global search capability in the problem space and can efficiently find optimal tour also it is quite simple and easy to apply, and it is efficient for large size matters. In this study a novel FA algorithm based on a hybrid mutation scheme (named MFA algorithm) was introduced for TSP. Experimental results show that this approach considers both running time and solution quality as well. In According to the results, it can be seen that the proposed algorithm has much better result compared to standard algorithms. The results of GA, PSO and FA are converged rapidly and there is no significant change by increasing the repetitions. The proposed algorithm is significantly improved by increasing the number of repetitions. As a future work, the algorithm FA Can be hybridized with SI algorithms to find better results.

**Acknowledgements** We thank all who have supported us for this research.

## References

1. Park YB (2001) A hybrid genetic algorithm for the vehicle scheduling problem with due times and time deadlines. Int J Productions Econom 73(2):175–188
2. Hu W, Liang H et al (2013) A hybrid chaos-particle swarm optimization algorithm for the vehicle routing problem with time window, 15, 1247–1270
3. Varunika A, Amit G, Vibhuti J (2014) An optimal solution to multiple travelling salesperson problem using modified genetic algorithm 3(1)
4. Chen S-M, Chien C-Y (2011) Parallelized genetic ant colony systems for solving the traveling salesman problem. Expert Syst Appl 38:3873–3883
5. Ray SS, Bandyopadhyay S, Pal SK (2004) New Operators of genetic algorithm for travelling salesman problem. In: Proceedings of the 17th international conference on pattern recognition, vol 2, pp 497–500
6. Yan, X, Zhang1, C (2012) An solve traveling salesman problem using particle swarm optimization algorithm 9(6), No 2
7. Shi XH, Liang YC, Lee HP, Lu C, Wang QX (2007) Particle swarm optimization-based algorithms for TSP and generalized TSP. Inform Process Lett 103:169–176
8. Marinakis Y, Marinaki M (2010) A Hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem. Comput Oper Res 37(3):432–442
9. Jie B (2012) A model induced max-min ant colony optimization for asymmetric traveling salesman problem. Appl Soft Comput 1365–1375
10. Saenphon T, Phimoltares S, Lursinsap C (2014) Combining new fast opposite gradient search with ant colony optimization for solving travelling salesman problem. Eng Appl Artif Intell 35:324–334
11. Mahi M, Kaan ÖB, 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
12. Nitesh MS, Chawda BV (2013) Memetic algorithm a metaheuristic approach to solve RTSP, IJCSEITR, ISSN 2249-6831, 3(2):183–186
13. George G, Raimond K (2013) Solving travelling salesman problem using variants of abc algorithm. Int J Comput Sci Appl (TIJCSA) 2(01) ISSN-2278-1080
14. Sobti S, Singla P (2013) Solving travelling salesman problem using artificial bee colony based approach. Int J Eng Res Technol (IJERT) 2(6)
15. Marinakis Y, Marinaki M, Dounias G (2011) Honey bees mating optimization algorithm for the Euclidean traveling salesman problem. Inf Sci 181(20)
16. Pathak N, Tiwari SP (2012) Travelling salesman problem using bee colony with SPV. Int J Soft Comput Eng (IJSCE) 2(3)
17. Saranya S, Vaijayanthi RP (2014) Traveling salesman problem solved using bio inspired algorithms (ABC). Int J Innovative Res Comput Commun Eng 2(1)
18. Kan J-M. Yi Z (2012) Application of an improved ant colony optimization on generalized traveling salesman problem. Energy Procedia 17(2012):319–325
19. Rizk-Allah RM, Elsayed M, Ahmed AE (2013) Hybridizing ant colony optimization with firefly algorithm for unconstrained optimization problems. Appl Math Comput 224:473–483
20. Kota L, Jarmai K (2013) Preliminary studies on the fixed destination MMTSP solved by discrete firefly algorithm. Adv Logis Sys 7(2):95–102
21. Yang XS (2009) Firefly algorithms for multimodal optimization. In: Stochastic algorithms: foundations and applications, SAGA 2009, Lecture notes in computer sciences, vol 5792, pp 169–178
22. http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/