13 Discrete Firefly Algorithm for Traveling Salesman Problem: A New Movement Scheme

Gilang Kusuma Jati[1], *Ruli Manurung*[1] *and Suyanto*[2]

[1]Faculty of Computer Science, Universitas Indonesia, Kampus UI, Depok, Jawa Barat, Indonesia; [2]Faculty of Informatics, Telkom School of Technology, Jl. Telekomunikasi No. 1, Terusan Buah Batu, Bandung, Jawa Barat, Indonesia

13.1 Introduction

The traveling salesman problem (TSP) is one of the most studied problems and has received growing attention in artificial intelligence, computational mathematics, and optimization theory since 1930. This problem was defined in the eighteenth century by Sir William Rowan Hamilton, a mathematician from Ireland, and by Thomas Penyngton Kirkman, a British mathematician. A detailed discussion about the work of Hamilton and Kirkman can be found in Biggs et al. (1986). The origins of the TSP are unclear, but it is believed that the general form of the TSP was first studied during the 1930s in Vienna and Harvard, most notably by a mathematician named Karl Menger who defined the problem, considered the obvious brute-force algorithm and observed the nonoptimality of the nearest neighbor heuristic. The problem was later introduced with the name TSP by Hassler Whitney and Merrill at Princeton University. A comprehensive overview about the connection between Menger and Whitney, and the development of the TSP can be found by Schrijver (1960).

The TSP can be described as follows: given a set of cities and known cost of travel (or distance) between each possible pairs, a salesman needs to find the best possible way of visiting all the cities exactly once and return back to the starting point that minimizes the travel cost (or travel distance). An exact solution to this problem involves algorithms that require seeking the possibility of all existing solutions, thus this problem belongs to the class of *nondeterministic polynomial time complete* (NP complete) problems, and it has been proven that there is no scheme or algorithm currently existing that can find the exact solution in polynomial time (Aurora et al.,

Swarm Intelligence and Bio-Inspired Computation. DOI: http://dx.doi.org/10.1016/B978-0-12-405163-8.00013-2

1992). As a result, the execution time complexity of this algorithm will be exponential to the size of the given input. If given n is the number of cities that will be visited, the total number of possible routes covering all cities can be declared as a set of feasible solutions of the TSP and is given as $(n-1)!/2$. A comprehensive overview of this line of research can be found by Gutin and Punnen (2002).

The TSP has several applications even in its purest formulation, such as planning, logistics, and the manufacture of microchips. A direct application of the TSP is drilling problem of printed circuit boards (PCBs) to connect a conductor on one layer with a conductor on another layer, or to position the pins of integrated circuits (Grötschel et al., 1991), overhauling gas turbine engines to guarantee a uniform gas flow through the turbines which have nozzle-guide vane assemblies located at each turbine stage (Plante et al., 1987), analysis of the structure of crystals in X-ray crystallography (Bland and Shallcross, 1989), connecting components on a computer board (Lenstra and Rinnooy Kan, 1975), material handling in a warehouse (Ratliff and Rosenthal, 1983), and vehicle routing problem. Beside the above examples, problems in real life like route distributions of transportation networks, planning of tourist routes, laying of pipelines needed for city planning and engineering construction are interlinked with the problems of finding the shortest route. Thus, it is very important to carry out research on the problem of the shortest route. The process of finding a solution for TSP plays an important role in real life, and it has attracted many scholars to do research on it.

The TSP is used as a benchmark for many optimization methods. Even though the problem is computationally difficult, many methods have been developed to solve TSP such as branch and bound (Land and Doig, 1960), Lin−Kernighan local search (Lin and Kernighan, 1973), heuristic search (Jiang et al., 2005), dynamic programming (Jellouli, 2001), and neural computing network (Abdel-Moetty, 2010). Evolutionary computation, especially metaheuristic algorithms, has shown very promising potential to solve optimization problems like TSP. It has been proven by many published methods to solve TSP, e.g., ant colony optimization by Marco Dorigo and Luca Maria Gambardella that mimic the movements of ants (Dorigo and Gambardella, 1996), genetic algorithm by Lanlan Kang that combined with ant colony optimization (Kang and Cao, 2010), and discrete particle swarm optimization by Matthias Hoffmann that used sequences of transpositions as the difference between tours (Hoffmann et al., 2011). Those researchers proposed methods that are combinations of a metaheuristic algorithm with a local search or other metaheuristic algorithms and focused on small TSP with hundreds of cities (nodes). Generally, accuracies of the methods are very high in that their produced solutions are very close to the known optimum solutions. In Tsai et al. (2004), the researchers focused on large TSP with thousands of cities. The method that they proposed, called HeSEA, is capable of solving a TSP of up to 3038 cities with a deviation of 0% compared to the known optimum solution. It can also solve a TSP of 13,509 cities with a deviation of only 0.74%.

The firefly algorithm (FA) is a nature-inspired metaheuristic optimization algorithm developed by Xin-She Yang that is inspired by the flashing behavior of fireflies (Yang, 2008), originally designed to solve continuous optimization problems

(Lukasik and Żak, 2010; Yang, 2009). However, the FA can be discretized to solve a permutation problem. The discrete firefly algorithm (DFA) has been successfully implemented to solve flow shop scheduling problems. The DFA outperforms existing algorithms such as ant colony algorithm (Sayadi et al., 2010). Recently, the evolutionary discrete firefly algorithm (EDFA) has been developed for solving TSP (Jati and Suyanto, 2011). However, fireflies in EDFA have no direction when moving. Hence, it moves using evolution strategies (ES) concept. So, each firefly will move using inversion mutation. One of the three primary rules in FA states that, for any two flashing fireflies, the less bright one will move toward the brighter one. When the firefly moves to another brighter firefly, the distance between them will decrease. However, the movement scheme for fireflies in EDFA violates these primary rules because the movement scheme does not guarantee that after one firefly moves toward the brighter one, the distance between them will decrease. In this chapter, a new movement scheme is proposed to solve the problem of this movement scheme in EDFA. This new movement scheme guarantees that after one firefly moves toward the brighter one, the distance between them will decrease.

13.2 Evolutionary Discrete Firefly Algorithm

The FA is a metaheuristic algorithm, inspired by the flashing behavior of fireflies. The primary purpose for a firefly's flash is to act as a signal system to attract other fireflies. Now this can idealize some of the flashing characteristics of fireflies so as to consequently develop firefly-inspired algorithms. By idealizing some of the flashing characteristics of fireflies, the firefly algorithm was developed by Yang (2008). FA uses the following three primary rules:

1. All fireflies are unisex which means that they are attracted to other fireflies regardless of their sex.
2. The degree of the attractiveness of a firefly is proportional to its brightness, thus for any two flashing fireflies, the less bright one will move toward the brighter one and their attractiveness will decrease as their distance increases. If there is no brighter one than a particular firefly, it will move randomly.
3. The brightness or light intensity of a firefly is affected or determined by the landscape of the objective function to be optimized.

For a maximization problem, the brightness can simply be proportional to the objective function. Other forms of brightness can be defined in a similar way to the fitness function in genetic algorithms or the bacterial foraging algorithm (Gazi and Passino, 2004).

13.2.1 The Representation of the Firefly

A solution representation for the TSP is a permutation representation as illustrated in Figure 13.1. Here, a firefly represents one solution. We adopt the so-called path representation, which is considered the most natural form of TSP representation for

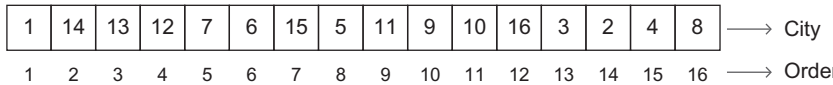

Figure 13.1 The permutation representation of a TSP solution.

chromosomes that represent individuals in genetic algorithms. In this representation, an element of an array represents a city (node) and the index represents the order of a tour. Two adjacent elements thus indicate an edge between two cities that lies along the route, e.g., see Figure 13.1, the edge between nodes 1 and 14 forms part of the route, since they occupy the first two positions in the array. We will use the notation $1-14$ to denote this edge.

13.2.2 Light Intensity

Light intensity is a value that represents the brightness of a firefly. This value depends on the total path distance of travel routes belonging to a firefly. Since the purpose of the TSP is to find a route with the minimum distance, a firefly which has less distance route will have a higher light intensity (brighter). The light intensity of a firefly x is calculated using Eqs. (13.1) and (13.2),

$$\text{distance\_route} = dist(x[1], x[n]) + \sum_{k=1}^{n-1} dist(x[k], x[k+1]) \tag{13.1}$$

$$I = \frac{1}{\text{distance\_route}} \tag{13.2}$$

where dist(i,j) is the Euclidean distance from city i to city j with $i, j \in [1,n]$, and n is the number of cities from the TSP problem data. With the above equation, the light intensity of a firefly will increase when the total path distance of travel routes belonging to a firefly decreases. In the FA, light intensity is used as an objective function.

13.2.3 Distance

In continuous optimization problems, the distance between two fireflies is simply calculated using Euclidian distance. For TSP, the distance between firefly i and firefly j can be defined as the number of different edges between them. In Figure 13.2, three edges $12-15$, $5-7$, and $6-11$ in firefly j do not exist in firefly i. Hence, the number of different edges between firefly i and firefly j is 3. Then, the distance between two fireflies is calculated using Eq. (13.3):

$$r = \frac{A}{n}.10 \tag{13.3}$$

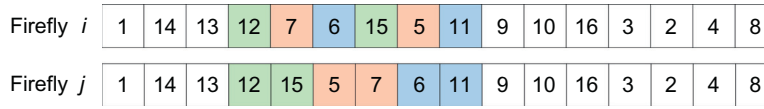

Figure 13.2 The distance between two fireflies i and j is defined as the number of different edges between them.

where r is the distance between any two fireflies, A is the total number of different edges between two fireflies, and n is the number of cities. This equation scales r in the interval [0,10][1] as r will be used for the attractiveness calculation.

13.2.4 Attractiveness

The attractiveness of a firefly is determined by its brightness, which is associated with the encoded objective function. In the original FA, the main form of the attractiveness function $\beta(r)$ can be any monotonic decreasing function:

$$\beta(r) = \beta_0 \, e^{-\gamma r^2} \tag{13.4}$$

where $\beta(r)$ is the attractiveness of a firefly when seen at distance r, r is the distance between two fireflies, β_0 is the brightness of a brighter firefly, and γ is a fixed light absorption coefficient. This scheme is completely adopted by the EDFA.

For any two fireflies, firefly i and another brighter firefly j, we first calculate the distance (r) between firefly i and firefly j by using Eq. (13.3). After that we calculate the attractiveness of firefly j when seen by firefly i at distance r by using Eq. (13.4). If the attractiveness of firefly j is greater than the brightness (light intensity) of firefly i, then firefly i will move toward firefly j, otherwise firefly i will move randomly.

13.2.5 Light Absorption

In essence, the light absorption coefficient γ characterizes the variation of the attractiveness value of a firefly. Its value is very important in determining the speed of convergence and how the FA behaves. In theory, $\gamma \in [0, \infty)$, but in practice γ is determined by the characteristics of the problem to be optimized.

In conditions where $\gamma \to 0$, the attractiveness will be constant and $\beta(r) = \beta_0$. In this case, the attractiveness of a firefly will not decrease when viewed by another. If $\gamma \to \infty$, this means the value of attractiveness of a firefly is close to zero when viewed by another firefly. It is equivalent to cases where the fireflies fly in a very foggy region randomly. No other fireflies can be seen, and each firefly roams in a

[1] The role of the constant multiplier here can be accounted for by the light absorption coefficient, γ (Section 13.2.5).

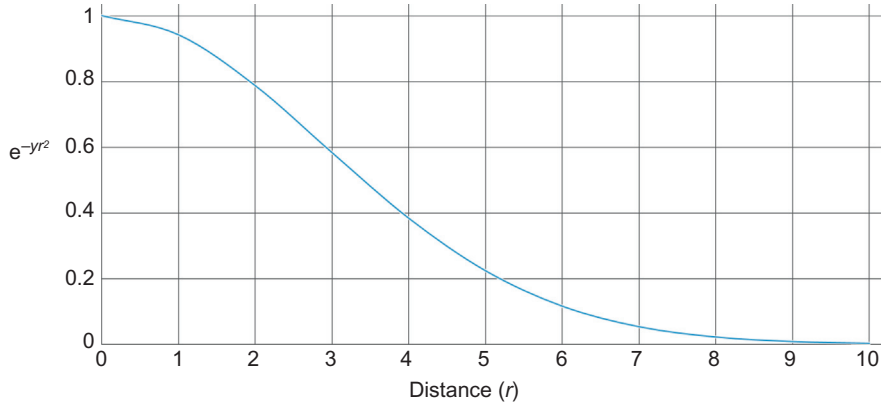

Figure 13.3 The correlation of distances and attractiveness with $\gamma = 0.06$.

completely random way. Therefore, this corresponds to a completely random search method.

The coefficient γ determines how much light intensity changes the attractiveness of a firefly over distances. In the EDFA, γ is in the interval [0.01,0.15] so that the attractiveness of a firefly viewed by the others will follow the curve shown in Figure 13.3.

This is the correlation between distances and attractiveness. The distance (r) scales in the interval [1,10] to handle the value of $e^{-\gamma r^2}$. If the distance is too high (more than 10), the value of $e^{-\gamma r^2}$ is close to zero, and the attractiveness of a firefly is also close to zero when viewed by another firefly, so the firefly will always move randomly.

13.2.6 Movement

The movement of a firefly i attracted to another brighter (more attractive) firefly j is determined by

$$x_i = \text{rand}(2, A) \tag{13.5}$$

where x_i is the step that must be taken by firefly i to move toward firefly j, and A is the total number of different edges between two fireflies i and j. The length of movement (step) of a firefly will be randomly selected from 2 to A using uniform distribution. When a firefly moves, existing solutions in the firefly are changed. Since the representation of a firefly is a permutation representation, we use inversion mutation to represent the movement.

Actually, a firefly in EDFA has no direction to move. Hence, it moves using ES concept. Each firefly will move using inversion mutation for m times. First, an index on the chromosome will be selected randomly, wherein an inversion mutation

is carried out. In other words, each firefly will have m new solutions. After p fireflies move and produce $p \times m$ new solutions, then the p best fireflies will be selected as the new population.

One of the three primary rules in FA states that for any two flashing fireflies, the less bright one will move toward the brighter one. When a firefly moves to another brighter firefly, the distance between them will decrease. This movement scheme violates these primary rules because the movement scheme does not guarantee that after one firefly moves toward the brighter one, the distance between them decreases.

13.2.7 Inversion Mutation

Inversion mutation is used to create new fireflies, i.e., new candidate solutions. First, randomly pick one point from the index of one chromosome (P_1), then select the second point (P_2) by summing the first point (P_1) and the length of movement. With inversion mutation, the path that has been formed can be maintained so the path formed previously is not damaged. Figure 13.4 shows an illustration of inversion mutation.

13.2.8 EDFA Scheme

The scheme of EDFA is illustrated by the following pseudocode. First, each firefly generates an initial solution randomly. For each firefly, find the brightest or the most attractive firefly. If there is a brighter firefly, then the less bright firefly will move toward the brighter one and if there is no brighter one, it will move randomly. When a firefly moves, the existing solution represented by the firefly is changed. Each firefly that moves randomly will generate a new solution using inversion mutation in m different positions on the chromosome. We refer to this constant m as the *updating index*. After p fireflies have moved, there will be $p \times m + 1$ fireflies at the end of an iteration since only the best firefly will be included in the selection process for the next iteration. Then, the p best fireflies will be chosen based on an objective function for the next iteration. This condition will continue until the maximum iteration is reached.

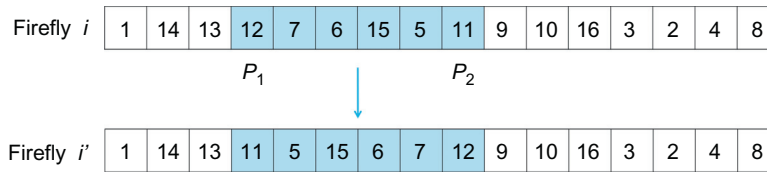

Figure 13.4 Inversion mutation of firefly with length of movement (step) = 5.

```
Input:
Define objective function f(x), population size p, light absorption coefficient γ, and updating
index m
Begin
        List<Firefly> temp = new List<Firefly>
        //initialize population of firefly
        Firefly[] x = new Firefly[p]
        for i = 1 to p do
                x[i] = Generate_Initial_Solution()
        endfor
        repeat
                for i = 1 to p do
                        Firefly f = Get_Most_Attractive_Firefly(x[i])
                        if (f != null) then
                                for j = 1 to m do
                                        Firefly fnew = Move_Firefly(x[i])
                                        temp.Add(fnew)
                                endfor
                        else
                                for j = 1 to m do
                                        Firefly fnew = Move_Random(x[i])
                                        temp.Add(fnew)
                                endfor
                        endif
                endfor
                //select p brightest fireflies from temp
                x = Get_Brightest_Fireflies(temp)
                temp.Clear()
        until stop condition true
        //output best firefly
        Output(x.min)
end
```

13.3 A New DFA for the TSP

Here, we discuss a new movement scheme called edge-based movement to guarantee that after one firefly moves toward the brighter one the distance between them will decrease. We also discuss the new DFA scheme.

13.3.1 Edge-Based Movement

As described in Section 13.2.6, the movement scheme for fireflies in EDFA violates the primary rules of FA because it does not guarantee that after one firefly moves toward a brighter one, the distance between them will decrease. Here we propose a new movement scheme that guarantees such a decrease.

The idea for the new movement begins from the rules that state the distance between two fireflies must be reduced after a firefly moves toward another brighter firefly. Since the distance between two fireflies is calculated using the total number of different edges between them, then when a firefly moves toward another brighter firefly, the total number of different edges between them must also be reduced. The total number of different edges between them can be reduced by adding an edge

that exists in the brighter firefly but does not exist in the less bright firefly in such a way that no similar edges between them that existed previously are removed.

The new movement scheme of DFA is illustrated by the following procedure. First, there are two fireflies, firefly i and another brighter firefly j. From all the edges that exist in firefly j but not in firefly i, randomly select one that will be added to firefly i, e.g., edge $x - y$ is the selected edge, where x and y are nodes or cities in the TSP. Find the positions of nodes x and y in firefly i. For each node, we then build a *segment* around it, i.e., a maximally long sequence of edges that contains edges that are also found in firefly j. The procedure to build this segment is simple. Starting from a node, which we consider to be a segment of length 1, we keep extending the segment to the left and right, adding edges that are found in firefly j, until it can no longer be extended. Let us call a segment built starting from node n as S_n, thus the segment built starting from node x is S_x, and the segment built starting from node y is S_y.

Once the two segments have been identified, we must merge them in such a way that nodes x and y appear adjacent to each other in firefly i, thus achieving the objective of adding edge $x - y$ to firefly i. There are four ways to merge these two segments, i.e., inserting S_x before S_y, inserting S_x after S_y, inserting S_y before S_x, and finally inserting S_y after S_x. Note that to ensure that edge $x - y$ appears in firefly i, some inversion of a segment might be required. Executing all four ways of merging these segments will yield four new fireflies (solutions).

Figures 13.5–13.7 illustrate this process through an example. Figure 13.5 shows two fireflies, firefly i and another brighter firefly j. The total number of different edges between them is seven, namely the edges 7 − 12, 12 − 13, 14-1, 1 − 8, 2 − 3, 10 − 15, and 5 − 6 that exist in firefly j but do not exist in firefly i.

Figure 13.6 shows the result of constructing segments in firefly i. The process begins by selecting one of the seven edges above that will be added to firefly i, say 2 − 3 is the selected edge. We then locate the position of node 2 (marked with x) and node 3 (marked with y) in the chromosome of firefly i. We then construct a segment (edge sequences) for each node. Starting from node 2, we extend the segment to the left to include 4-2 and 8-4, since both these edges are also found in firefly j. However, we stop at node 8 since 7 − 8 is not found in firefly j. Attempting to extend this segment to the right fails because the edge 2-1 is also not found in firefly j. Likewise for node 3, we extend the segment to the right to include 3 − 16, 16-11, 11-9, and 9 − 10, since all of these edges are also found in firefly j. However, we stop at node 10 since 10 − 13 is not found in firefly j. Attempting to extend this segment to the left fails because the edge 12-3 is also not found in firefly j. In fact, it is guaranteed that each segment can only be extended

Firefly i	6	7	8	4	2	1	5	15	12	3	16	11	9	10	13	14
Firefly j	6	7	12	13	14	1	8	4	2	3	16	11	9	10	15	5

Figure 13.5 Two fireflies with seven different edges.

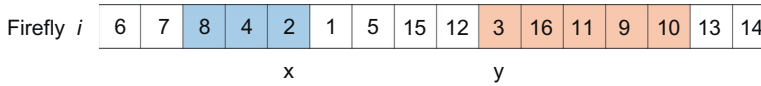

Figure 13.6 Firefly i with the segments of each node in the selected edge.

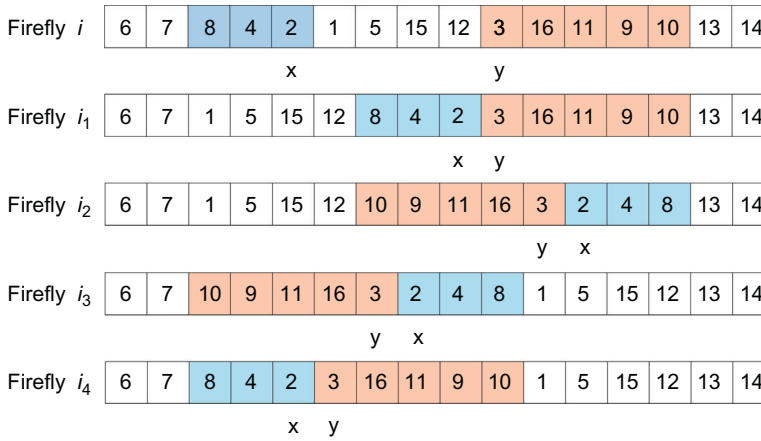

Figure 13.7 Merging process of two segments in firefly.

in one direction, since the aim is to add the edge $x - y$ into firefly i, thus $x - y$ must not previously exist. Thus, S_x is 8-4-2 and S_y is 3-16-11-9-10.

Finally, the merging process of S_x and S_y is illustrated in Figure 13.7. There are four ways to merge these two segments, i.e., inserting S_x before S_y, inserting S_x after S_y, inserting S_y before S_x, and finally inserting S_y after S_x. Fireflies i_1', i_2', i_3', and i_4' show the results of applying these four possibilities, yielding four new fireflies (solutions). Note that for fireflies i_2' and i_4', the segments S_x and S_y had to be inverted to ensure that edge $x - y$ can be added.

Note that in all of these solutions, edge $2 - 3$ has been successfully added to firefly i and has been done in such a way that no other edge belonging to firefly i that also appears in firefly j has been removed as a result. Thus, if distance is measured in terms of the number of different edges between two fireflies, the distance between them has now been decreased.

13.3.2 New DFA Scheme

The scheme of the new DFA, which employs the new edge-based movement described in Section 13.3.1, is illustrated by the following pseudocode. The concept is similar to EDFA, but there are some differences. In the begining, each firefly

will generate an initial solution randomly. For each firefly, we find the brightest or the most attractive firefly. If there is a brighter firefly, then the less bright firefly will move toward the brighter one with the new movement scheme and if there is no brighter one, it will move randomly (move with ES concept like in EDFA). When a firefly moves, the existing solution represented by the firefly is changed. Each firefly that moves randomly will generate a new solution using inversion mutation in m different positions, i.e., the updating index, on the chromosome. Each firefly that moves toward a brighter firefly will yield four new solutions, and one solution is chosen randomly. After p fireflies have moved, the total number of fireflies in EDFA will be constant at each generation, but in the new DFA it depends on how many fireflies move randomly and how many move toward the others. For the next iteration, the p best fireflies will be chosen based on an objective function. This condition will continue until the maximum iteration is reached.

```
Input:
Define objective function f(x), population size p ,light absorption coefficient γ, and updating
index m
Begin
        List<Firefly> temp = new List<Firefly>
        //initialize population of firefly
        Firefly[] x = new Firefly[p]
        for i = 1 to p do
                x[i] = Generate_Initial_Solution()
        endfor
        repeat
                for i = 1 to p do
                        Firefly f = Get_Most_Attractive_Firefly(x[i])
                        if (f != null) then
                                Firefly[] newSolutions = new Firefly[4]
                                newSolutions = Edge-based_Movement(x[i])
                                Firefly fnew = newSolution[Random(1,4)]
                                temp.Add(fnew)
                        else
                                for j = 1 to m do
                                        Firefly fnew = Move_Random(x[i])
                                        temp.Add(fnew)
                                endfor
                endif
                endfor
                //select p brightest fireflies from temp
                x = Get_Brightest_Fireflies(temp)
                temp.Clear()
        until stop condition true
        //output best firefly
        Output(x.min)
end
```

13.4 Result and Discussion

In this chapter, the new DFA is applied to 7 TSP instances obtained from TSPLIB (2011). Table 13.1 lists the problem names, numbers of cities, and the lengths of their known optimal tours. The types of TSP instances in TSPLIB (2011) are based

Table 13.1 Summary of 7 TSPs Taken from TSPLIB: Problem Names, Number of Cities (nodes), and the Length of the Optimal Tour

Problem Names	Number of Cities	Length of the Optimal Tour
ulysses16	16	74.109
ulysses22	22	75.665
gr202	202	549.998
tsp225	225	3859
a280	280	2586.769
pcb442	442	50783.547
gr666	666	3952.535

on Euclidian distances, wherein a TSP instance provides some cities with their coordinates. The number in the name of an instance represents the number of provided cities. For example, ulysses16 provides 16 cities with their coordinates. The problem is thus what is the best tour to visit the 16 cities, according to their Euclidian distances, with a condition where each city should be visited only once.

13.4.1 Firefly Population

Population size (p) critically determines the computation time. Here, the new DFA is tested using various population sizes on problem gr202 to investigate its correlation with the number of trials needed by new DFA to obtain the optimum solution. Table 13.2 represents the correlation of the population size with the average trials to reach the optimum solution (with accuracy of 100%). The results reported are averaged after running the experiment 50 times. In new DFA, the average trials decrease when the population size is increased from 2 to 5. But the average trials increase when the population size is 6 or more. Thus, an increasingly large population does not guarantee that FA will reach an optimum solution more quickly. According to Table 13.2, the best population size for new DFA given problem gr202 is 5.

13.4.2 Effect of Light Absorption

The light absorption coefficient (γ) also critically determines the computation time. Various values of light absorption were tested on problem gr202 to evaluate its correlation with the number of trials needed by new DFA to obtain the optimum solution. Table 13.3 illustrates the relationship between the light absorption and the average trials to obtain the optimum solution. The results reported are averaged after running the experiment 50 times. The light absorption coefficient is varied from 0.0001 to 0.21, yielding different average trials. Thus, the light absorption coefficient significantly affects the average trials of new DFA. In new DFA, the average running time decreases when the light absorption coefficient is increased.

Table 13.2 Correlation of the Population Size and the Average Trials Needed by New DFA to Obtain the Optimum Solution

Population Size	Average Trials	Standard Deviation	Average Running Time(s)
2	110,617	51,036	1.651
3	84,110	67,377	1.615
4	59,478	8156	1.426
5	**55,690**	**13,814**	**1.045**
6	58,700	12,755	1.904
7	61,395	22,795	2.593
8	59,761	14,768	2.690
9	71,086	34,471	3.512
10	63,240	14,227	3.583
11	77,126	39,328	4.990
12	58,537	13,095	4.161
13	68,482	36,616	5.203
14	57,359	16,225	4.763
15	59,176	9392	5.328
16	58,489	17,302	5.609
17	68,300	18,489	7.025
18	70,667	14,951	7.390
19	66,508	13,434	8.038
20	61,912	6963	8.455

Note: The bold values represent the best result for all setting parameters. The best result is the minimum value in the column of average trials.

There are two types of firefly movement. The first is where a firefly deliberately moves toward a brighter one (marked as "move toward" in Table 13.3), and the second is a random movement (marked as "move random" in Table 13.3). The number of random moves increases when the light absorption coefficient is increased. This corresponds to the theory that if $\gamma \to \infty$, the value of attractiveness of a firefly is close to zero when viewed by another firefly. It is equivalent to cases where the fireflies fly in a very foggy region randomly. No other fireflies can be seen, and each firefly roams in a completely random way. When $\gamma \geq 0.2$, FA behaves like random search.

13.4.3 Number of Updating Index

The number of updating index (m) also determines the computation time. New DFA is tested using various number of updating index on problem gr202 to investigate its correlation with number of trials needed by new DFA to obtain the optimum solution. Table 13.4 represents the correlation of the number of updating index with the average trials to reach the optimum solution (with accuracy of 100%). The results reported are averaged after running the experiment 50 times.

Table 13.3 Correlation of the Light Absorption Coefficient (γ) and the Average Trials Needed by New DFA to Obtain the Optimum Solution

Light Absorption	Average Move Toward	Average Move Random	Average Trials	Standard Deviation	Average Running Time (s)
0	4566	61,903	66,469	22,776	1.963
0.0001	4382	61,467	65,849	18,438	1.988
0.0002	4349	61,441	65,790	25,685	1.899
0.0003	4366	62,442	66,808	18,993	1.979
0.0004	4152	58,233	62,385	16,805	1.808
0.0005	4258	61,572	65,830	16,309	1.947
0.0006	4350	64,451	68,801	22,867	1.901
0.0007	4115	60,008	64,193	17,321	1.835
0.0008	4276	64,245	68,251	17,397	1.869
0.0009	4046	59,626	63,671	16,139	1.764
0.0010	3961	58,656	62,617	18,608	1.703
0.0020	3704	59,572	63,276	18,214	1.721
0.0030	3379	58,266	61,645	24,869	1.519
0.0040	3325	64,982	68,307	29,716	1.620
0.0050	2757	56,866	59,623	14,037	1.456
0.0060	2728	61,684	64,412	15,617	1.439
0.0070	**2629**	**53,061**	**55,690**	**13,814**	**1.045**
0.0080	2454	67,046	69,500	24,023	1.448
0.0090	2137	62,950	65,807	16,222	1.292
0.0100	1961	63,918	65,879	23,338	1.276
0.0200	886	65,728	66,614	12,555	0.985
0.0300	474	71,377	71,852	15,209	1.024
0.0400	250	75,881	76,131	14,699	0.992
0.0500	129	80,370	80,499	28,774	1.008
0.0600	50	77,086	77,136	13,316	0.989
0.0700	24	83,680	83,705	21,995	1.030
0.0800	9	77,719	77,728	16,071	0.967
0.0900	5	85,464	85,470	27,727	1.060
0.1000	3	79,137	79,140	21,252	1.249
0.1100	3	82,104	82,107	17,631	1.001
0.1200	2	82,545	82,547	17,891	1.004
0.1300	2	83,767	83,769	18,050	1.024
0.1400	1	84,612	84,613	27,421	1.019
0.1500	1	81,360	81,361	19,087	1.006
0.1600	1	84,644	84,645	21,999	0.945
0.1700	1	82,227	82,228	16,755	0.917
0.1800	1	79,272	79,273	22,978	0.903
0.1900	1	82,305	82,305	19,994	0.942
0.2000	0	80,600	80,600	15,834	0.904
0.2100	0	78,786	78,786	14,881	0.884

Note: The bold values represent the best result for all setting parameters. The best result is the minimum value in the column of average trials.

Table 13.4 Correlation of the Number of Updating Index (m) and the Average Trials Needed by New DFA to Obtain the Optimum Solution

Number of Updating Index	Average Trials	Standard Deviation	Average Running Time (s)
1	300,514	39,698	5.687
2	67,258	13,404	1.645
3	61,181	20,010	1.679
4	61,854	21,126	1.520
5	63,683	24,264	1.444
6	57,521	13,848	0.699
7	58,128	14,083	0.668
8	60,605	25,405	1.016
9	63,226	37,266	1.190
10	60,629	21,925	0.698
11	**55,690**	**13,814**	**1.045**
12	62,638	18,734	0.532
13	63,533	21,430	1.529
14	69,212	60,871	1.361
15	68,964	63,109	1.568
16	60,572	13,144	0.994

Note: The bold values represent the best result for all setting parameters. The best result is the minimum value in the column of average trials.

In new DFA, the average trials decrease when the number of updating index is increased from 1 to 11. But the average trials become unstable when the number of updating index is 12 or more. A large number of updating index does not guarantee that FA will reach an optimal solution more quickly. According to Table 13.4, the best number of updating index for new DFA for problem gr202 is 11.

13.4.4 Performance of New DFA

New DFA is examined for 7 TSP instances detailed in Table 13.1 to compare its performance with EDFA proposed in Jati and Suyanto (2011). Here, new DFA uses a population size of 5 fireflies and the light absorption coefficient is 0.001. The accuracy is calculated by Eq. (13.6). New DFA is implemented using Visual C#. Net. The results reported are averaged after running the experiment 50 times, the same as for EDFA studied by Jati and Suyanto (2011). The results are summarized in Table 13.5. New DFA obtains the best solution for ulysses16, ulysses22, gr202, and gr666 for all runs. New DFA is not optimal for three other instances: tsp225, a280, and pcb442, but its accuracy is slightly better than that of EDFA. For all seven instances, new DFA runs much faster than EDFA. The average speedup

Table 13.5 The Comparison Between New DFA and EDFA

Problem Names	Best Solution Known	EDFA			New DFA			Speedup Factor
		Average Accuracy	Average Trials	Average Time	Average Accuracy	Average Trials	Average Time	
ulysses16	74.10	100.11	25,160	0.41	100.09	1149	0.01	21.89 ×
ulysses22	75.66	100.20	235,720	6.59	100.18	8987	0.02	26.22 ×
gr202	549.99	100.47	1,443,372	51.16	100.17	55,690	1.04	25.91 ×
tsp225	3859.00	88.33	2,739,000	412.27	88.49	445,398	120.42	6.14 ×
a280	2586.76	88.29	3,592,440	691.88	88.38	548,930	251.76	6.54 ×
pcb442	50,783.54	88.50	8,196,960	3404.21	88.53	1,187,428	167.23	6.90 ×
gr666	3952.53	100.03	1,649,560	393.02	100.08	341,519	51.33	4.83 ×
							Average:	14.06 ×

factor is 14.06 times, where the speedup factor is determined as the ratio between the average trials of the new DFA and EDFA:

$$\text{accuracy} = \frac{\text{best\_solution\_known}}{\text{best\_solution\_found}}.100\% \tag{13.6}$$

13.5 Conclusion

A new proposed movement scheme, called edge-based movement, works very well in new DFA. Our comparative testing to EDFA shows that the new movement in new DFA produces slightly better accuracy with much faster average time, which in our experiments yielded an average speedup factor of 14.06. Although the new movement guarantees to make an individual closer to brighter one, it does not guarantee to improve the fitness of that individual. Hence, the new DFA should be designed to have an appropriate light absorption so that it has some random movements to make the evolution works well.

Acknowledgment

Many thanks to all colleagues at Universitas Indonesia and Telkom School of Technology for the support given.

References

Abdel-Moetty, S.M., 2010. Traveling salesman problem using neural network techniques. Proceedings of the Seventh International Conference on Informatics and Systems (INFOS). IEEE Computer Society, Washington, DC, pp. 1–6. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5461754&pcontentType=Conference + Publications&searchField%3DSearch_All%26queryText%3DTraveling + Salesman + Problem + Using + Neural + Network + Techniques>.

Aurora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M., 1992. Proof verification and hardness of approximation problems. In: Proceedings of the 33th IEEE Symposium Foundations of Computer Science. IEEE Press, Pittsburgh, PA, pp. 10–22.

Biggs, N.L., Lloyd, E.K., Wilson, R.J., 1986. Graph Theory 1736-1936. Clarendon Press, Oxford (ISBN 978-0-19-853916-2).

Bland, R.E., Shallcross, D.E., 1989. Large traveling salesman problem arising from experiments in X-ray crystallography: a preliminary report on computation. Oper. Res. Lett. 8 (3), 125–128.

Dorigo, M., Gambardella, L.M., 1996. Ant Colonies for the Traveling Salesman Problem. University Libre de Bruxelles, Belgium.

Gazi, K., Passino, K.M., 2004. Stability analysis of social foraging swarms. IEEE Trans. Syst. Man Cybern. Part B—Cybern. 34, 539−555.

Grötschel, M., Jünger, M., Reinelt, G., 1991. Optimal control of plotting and drilling machines: a case study. Math. Methods Oper. Res. 35 (1), 61−84.

Gutin, G., Punnen, A.P., 2002. The traveling salesman problem and its variations, Comb Optim, vol. 12. Springer-Verlag, Berlin, Heidelberg, pp. 1−28.

Hoffmann, M., Mühlenthaler, M., Helwig, S., Wanka, R., 2011. Discrete particle swarm optimization for TSP: theoretical result and experimental evaluations, International Conference on Adaptive and Intelligent System (ICAIS 2011), vol. 6943. Springer-Verlag, Berlin, Heidelberg, Lecture Notes in Artificial Intelligence (LNAI), pp. 416−427.

Jati, G.K., Suyanto, 2011. Evolutionary discrete firefly algorithm for travelling salesman problem, Second International Conference on Adaptive and Intelligent System (ICAIS 2011), vol. 6943. Springer-Verlag, Berlin, Heidelberg, Lecture Notes in Artificial Intelligence (LNAI), pp. 393−403.

Jellouli, O., 2001. Intelligent dynamic programming for the generalized traveling salesman problem. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, vol. 4. IEEE Computer Society, Washington, DC, pp. 2765-2768.

Jiang, H., Zhou, Z., Zhou, P., Chen, G.L., 2005. Union search: a new metaheuristic algorithm to the traveling salesman problem. J. Univ. Sci. Technol. China. 35, 367−375.

Kang, L., Cao, W., 2010. An improved genetic and ant colony optimization algorithm for travelling salesman problem. Third International Symposium on Information Science and Engineering (ISISE 2010). IEEE Computer Society, Washington, DC, pp. 498−502.

Land, A., Doig, A., 1960. An automatic method of solving discrete programming problems. Econometrica. 28 (3), 497−520.

Lenstra, J.K., Rinnooy Kan, A.H.G., 1975. Some simple applications of the traveling salesman problem. Oper. Res. Q. 26, 717−733.

Lin, S., Kernighan, B., 1973. An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. 21 (2), 498−516.

Lukasik, S., Żak, S., 2010. Firefly algorithm for continuous constrained optimization tasks. Syst. Res. Inst. Pol. Acad. Sci.1−10.

Plante, R.D., Lowe, T.J., Chandrasekaran., R., 1987. The product matrix traveling salesman problem: an application and solution heuristics. Oper. Res. 35, 772−783.

Ratliff, H.D., Rosenthal, A.S., 1983. Order-picking in a rectangular warehouse: a solvable case for the travelling salesman problem. Oper. Res. 31, 507−521.

Sayadi, M.K., Ramezanian, R., Ghaffari-Nasab, N., 2010. A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems. Int. J. Ind. Eng. Comput.1−10.

Schrijver, A., 1960. On the history of combinatorial optimization. In: Aardal, K., Nemhauser, G.L., Weismantel, R. (Eds.), Handbook of Discrete Optimization. Elsevier, Amsterdam, pp. 1−68.

Tsai, H.K., Yang, J.M., Tsai, F.Y., Kao, C.Y., 2004. An evolutionary algorithm for large traveling salesman problems. IEEE Transactions on Systems, Man, and Cybernetics, Part B. IEEE Computer Society, Washington, DC, pp. 1718−1729.

TSPLIB95: Ruprecht—Karls—Universitat Heildelberg, 2011. <http://www.iwr.uini-heidelberg.de/groups/comopt/software/TSPLIB95/>. (accessed 09.17.12).

Yang, X.S., 2008. Nature-Inspired Metaheuristic Algorithm. Luniver Press, UK.

Yang, X.S., 2009. Firefly algorithm for multimodal optimization, Proceedings of the Fifth International Conference on Stochastic Algorithms: Foundations and Applications (SAGA'09), vol. 5792. Springer-Verlag, Berlin, Heidelberg, Lecture Notes in Computer Science (LNCS), pp. 169−178.