A Novel Hybrid Genetic Algorithm based Firefly Mating Algorithm for Solving TSP

Sunanda Jana[1], Rajrupa Metia[1]
1(Computer Science and Engineering, Haldia Institute of Technology, India)

ABSTRACT
TSP is an NP-complete based mathematical problem, which has enormous applications in the field of vehicle routing problems, logistics, planning and scheduling etc. The traveling salesman problem (TSP) is a problem in combinatorial optimization. Several heuristics are there to solve this interesting structure. One of the heuristics, genetic algorithm (GA) is used by many researchers to solve TSP effectively, but they face various problems. GA has so many lacunas, and to overcome these, we have hybridized GA in a novel way. In this paper, we have developed a hybrid genetic algorithm based firefly mating algorithm (HGFMA), which can solve TSP instances with a greater success rate for easy, medium, and hard difficulty level based on number of cities. Our proposed method has controlled "getting stuck in local optima," considering less population and less generation.
Keywords - Genetic Algorithm, Firefly Mating Algorithm, Fitness function, Mating capability, Female pheromones, Male flash brightness, Crossover, TSP, Mutation, Population, Chromosomes

I. INTRODUCTION

Given a collection of cities and the distance of travel between each pair of them, the traveling salesman problem is to find the shortest way of visiting all of the cities and returning to the starting point. Though the statement is simple to state but it is more difficult to solve. Traveling Salesman Problem [8][9] is an optimization problem and has a vast search space and is said to be NP-hard, which means it cannot be solved in polynomial time. It is one of the most fundamental problems in the field computer science in today's time. The Traveling salesman problem is being applied in many fields nowadays. Some of its applications are vehicle routing, manufacturing of microchips, packet routing in GSM, drilling in printed circuit boards etc. In simpler words, say we have a set of n number of cities, and then we can obtain (n - 1)! alternative routes for covering all the n cities. Traveling salesman problem is to procure the route which has the least distance. Given a set of cities and distances between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point. Note the difference between Hamiltonian Cycle and TSP. The Hamiltonian cycle problem is to find if there exists a tour that visits every city exactly once. Here we know that Hamiltonian Tour exists (because the graph is complete) and in fact, many such tours exist, the problem is to find a minimum weight Hamiltonian Cycle. For example, consider the graph shown in the figure on the right side. A TSP tour in the graph is 1-2-4-3-1. The cost of the tour is 10+25+30+15 which is

80.The problem is a famous NP-hard problem. There is no polynomial-time known solution for this problem.

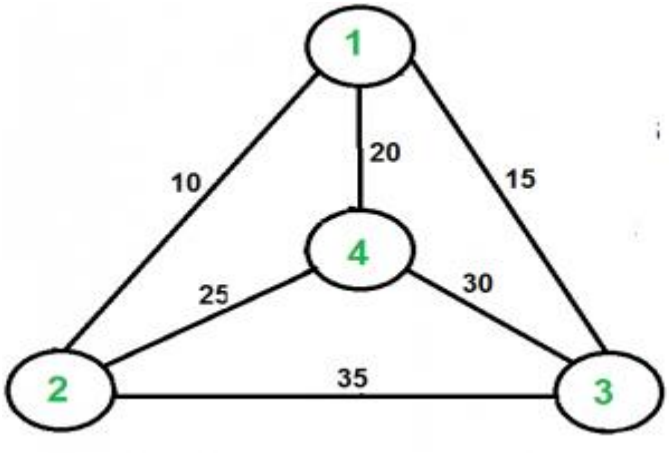

Figure 1:A given TSP consists of 4 cities.

Nature always inspired us to solve complex problems by providing several heuristic algorithms which can be applied to solve difficult combinational problems, like TSP problem. Several techniques have already been developed to solve this interesting logical structure, and GA [1] is extremely popular among them

As it is observed that the major insufficiencies associated with the use of GAs are the premature convergence [6] to solutions in implementing local optima of the objective function. The problem is highly related to the GA populations' loss of genetic diversity. This situation decreases the efficiency of computing solutions. In this article, the aim of our work is to develop some of the techniques to control the premature convergence to local optima.

To sort out the problems, our effort anticipated a new hybrid GA by embedding firefly mating algorithm into it to create a revolution in the field of developing efficient and effective Sudoku solver. We all know that solving TSP is not only fun but also a challenge. The novelty of our proposed method is that considering less population and less generation TSP instances can be solved with a greater success rate for easy, medium, and hard difficulty level with controlled "stuck into local optima".

In this novel hybrid approach, we have applied a relatively new heuristic algorithm, i.e., Firefly Mating Algorithm over GA to solve TSP of all levels of difficulty. Our proposed hybrid method works effectively to avoid the loss of genetic diversity of the whole population, and in principle, will not damage the convergence process.

Our paper is organized as follows: The introduction and literature survey of TSP are briefly described in Sections 1 and 2, respectively. Section 3 comprises three subsections: in Section 3.1, the firefly mating algorithm is briefly introduced; the proposed algorithm is described in Section

3.2; and in Section 3.3, an exhaustive study of our proposed algorithm is presented. All experimental results have been discussed in Section 4. Finally, conclusions and future works are discussed in Section 5.

II. LITERATURE REVIEW

Many natures' inspired metaheuristic algorithms for solving critical, tricky TSP problems have been proposed by many researchers. Evolutionary algorithms (like Genetic Algorithm or GA, Firefly Mating Algorithm or FMA) are mostly pertained to crack an NP-complete problem [13], and TSP problem is also an NP-complete problem [1]. Other than soft computing methods, the brute force algorithm is also a famous one, but the main disadvantages of this approach are that it uses complex logic, requires multiple function calls, the execution demands huge recursion which can lead to the insufficient memory problem since the CPU stack is limited in size and it can scarcely be optimized. Thus, human intervention is necessary to reduce the memory space, whereas, in our hybrid GA, there is no need for recursion; hence, the CPU stack utilization is decreased and requires less computation.

According to some researcher, by considering different evolutionary techniques, they stated that the performance is effective and efficient for TSP, but for hard onward the success rate decreases. In [3], the authors have covered multiple combinations of crossover and mutations for solving TSP. Among them, the random mutations with uniform crossover have more success rate, whereas the two-point crossover is effective in the speed of convergence.

Nowadays, a hybridization approach is primarily embedded to overcome the limitations of GA in terms of local searching capability along with prematurity in case of convergence to make the algorithm more effective and robust. In this paper, firefly mating algorithm (FMA) [5], which is a swarm intelligence algorithm, is incorporated in GA to develop a new hybrid GA where GA is the core of the algorithm. Till now, no such work has been implemented in the field of solving TSP using this new metaheuristic firefly mating algorithm where GA is the backbone of the implementation.

III.PROPOSED WORK

We know that there are several heuristic evolutionary algorithms [1] like Genetic Algorithms (GA) [1], Ant Colony Optimization (ACO) [1], and Artificial Bee Colony Optimization (ABCO) [1]. In this paper, we have implemented a new hybrid approach by applying a relatively new heuristic algorithm, i.e., Firefly Algorithm over GA to solve TSP.

The firefly mating algorithm (FMA) is originated by getting inspired by the mating structure of fireflies from nature. The FMA algorithm has three main stages:

- Female releases pheromone, and based on that male selects the female.
- A female selects male for mating according to the brightness of the flash.
- Both male and female mate repeatedly until they become incapable of further mating.

The first step of the algorithm is an initialization, where the number of fireflies, their mating capability, and their fitness is initialized or calculated. After the initialization phase, the next

crucial phase of this algorithm is selecting pairs for performing mating. Normally, in the genetic algorithm, the chromosomes are selected by the Roulette Wheel or the tournament selection technique, but the FMA algorithm uses a natural concept on how fireflies mate depending on the stated three main stages above, i.e., fireflies mate depending on the mutual attraction. Thus, this approach helps the algorithm to overcome the local optima problem [5].

In our hybrid Genetic Algorithm based Firefly Mating Algorithm (HGFMA), the core algorithm is the Genetic Algorithm (GA). GA is the class of evolutionary algorithm deduced by Goldberg based on Darwin's Theory of Evolution, i.e., "survival of the fittest". Our algorithm uses Firefly over GA to develop this hybrid algorithm to overcome the problems of GA. The fitness function determines how near an individual to its best solution is. The selector function selects the fittest individual for the population of the next generation. Eventually, this process continues until the population converges to an optimal or near-optimal solution.

In our algorithm, the firefly population is a two-dimensional (2D) matrix of size $p \times 81$, where p is the size of the population, and an array of size 81 denotes each firefly. Each gene of the firefly can take value from 1 to 9. Initialization is the first phase of our algorithm, where the firefly population is created. Before moving to the crossover phase, the compatibility of the firefly mating pairs is checked in the selection phase and accordingly the best compatible firefly pairs are selected for crossover to avoid redundant mating. Crossover is the first phase of mating and reproduction, where after mating parents produce offspring. The mutation is the immediate step after crossover, and it sets some random alterations in the structure of the gene of the offspring and maintains genetic diversity from parent to offspring. In our work, we have modelled the algorithm in a simple but innovative way such that it can reduce the number of generations and minimizes the error value at each step by considering some constraints. In the initialization phase, by opting for no duplication in a city reduces the error; in the selection phase, the redundant crossover is omitted, and again, in the mutation phase, the neighbourhood-based mutation process eliminates the duplication from rows and columns. The detailed algorithm has been stated below.

A. Proposed Algorithm

In our proposed algorithm, we have implemented a Hybrid TSP solver by embedding the concept of firefly mating algorithm (FMA) on GA by considering 9 cities. The FMA algorithm is a nature-based swarm optimization algorithm. Here, the algorithm is incorporated in the selection phase before crossover along with the neighbourhood mutation process, termed as a Hybrid Genetic Algorithm based Firefly Mating Algorithm for Solving TSP (HGFMA). The focal aim of the HGFMA is to select pairs of appropriate parents for mating and undergo crossover. This technique guarantees that compatible firefly pairs will get selected and can mate multiple times depending on their capability.

Hybrid Genetic Algorithm based Firefly Mating Algorithm (HGFMA) for solving TSP

Begin

Initialize the firefly Population_size, Maximum_generation, Mutation_probability, table_top.

Set the fitness function

```
For all fireflies do                                              /* Initialization*/
    Assign empty cells of each firefly with a random number from 1 to 9 without any
      repetitions in the subgrid.
      Compute the total mating pair (MP)
While gen = 1 to Maximum_generation or converged (whichever is earlier) do
      Consider half of the firefly as Male and rest half as Female.
      Calculate the fitness of both Male and Female firefly.      /*Selection Technique*/
    Compute female pheromones (Ph) for each Female to Male firefly pair.
    Compute male flash brightness (β) for each Male to Female firefly pair.
    Compute mutual attraction (MuA) for each firefly pair and store in an array.
    Set the Selection_table.
      Sort the Selection_table according to Mutual_Attraction (MuA) in descending order.
    Compute Mating_capability (MCap) for each firefly.
      For cross = 1 to Population_size/2 do                       /* Crossover*/
        Select a pair of fireflies from the Selection_table pointed by table_top as parents.
        If both the selected fireflies have MCap > 0
        Set crossover_point in parents after three sub-grids and swap alternate por-
          tions to form new offspring
          Decrease table_top  and MCap by 1.
    For i = 1 to offspring_size do                                /*Mutation*/
        For all subgrids do
          If the generated random probability < Mutation_probability and
            Parents(t) = Select_Parents(P(t));
            Offspring(t) = Procreate(P(t));
            p(t+1) = Select_Survivors(P(t), Offspring(t));
            t = t + 1;
                        Select the chromosomes if fitness is increased after crossover.
  Combine the parent and offspring population and sort according to fitness and select al
    ternate chromosomes for next generation.                      /*Selection for next generation*/
End
```

B. Exhaustive study of Proposed Algorithm

This section explains our proposed algorithm, stepwise.

Initialization

The proposed Hybrid Genetic Algorithm based Firefly Mating Algorithm for Solving TSP (HGFMA) begins by the initialization phase. Here a collection of fireflies refers to as a population. The TSP is represented as "firefly" by an array of size 81 provided with a combination of has to find the shortest route to travel through all the cities back to the city 1. A chromosome representing the path chosen can be represented as: 1 to 9. In this hybrid algorithm, each chromosome of the genetic algorithm is represented as fireflies. Hence, to represent chromosome, onwards, we use the term firefly. We keep a copy of firefly to track the given clue

location. The parent population is represented by a two-dimensional array where each row represents a firefly. Lastly, the maximum generation, epoch, the mutation probability, etc., are initialized.

Calculation of Fitness Function and Total Mating Pair

Fitness Function

The fitness function is used to compute the fitness of the chromosome. The proposed fitness function is modelled in a simple way, i.e., The fitness function is built based on the idea of frequency calculation of an integer in an array. For example: In an array [4,3,2,4,4,3] frequency of 4 is 3, frequency of 3 is 2, frequency of 2 is 1. Thus, for the occurrence of each integer if frequency exceeds 1, then the calculated error value is incremented by 1. This process is repeated for both row and column. The total error calculation is done by performing the summation of row error and column error considering city error as zero since no repetition is allowed in a city and the total error is subtracted from 100 to get the amount of fitness as shown in equation (1). In our described method, the chromosome with fitness value 100 is assumed as the best individual.

$$\text{Fitness Value (F)} = 100 - \left(\sum_{y=1}^{9} (freqr(y) - 1) + \sum_{y=1}^{9} (freqc(y) - 1) \right) \quad (1)$$

In the above equation, (freqr(y)–1) denotes the row error, (freqc(y)–1) denotes the column error, y represents an integer between 1 and 9, and F represents the fitness value.

Total Mating Pair (MP)

For a firefly population of size m, we can consider m/2 female and m/2 male fireflies for our convenience. Now each female can mate with every male firefly, and each male can mate with all other female fireflies, as shown in Figure 2. Let us assume N = m/2.

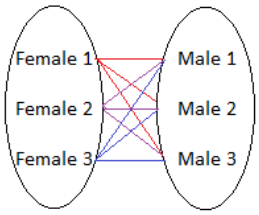

Figure 1. Pairing structure of fireflies during the mating process.

Hence, the total mating pair (MP) generated is shown in equation (2).

$$MP = N^2 \quad (2)$$

Thus, we must generate the compatibility for MP number of mating pairs before selecting any particular pair. Note that, in our proposed algorithm, we have considered the population size as even; thus, the numbers of female and male fireflies are always equal.

Selection Technique of Parent for Mating

This step primarily implies the logic behind the FMA algorithm for selecting the perfect mating pair. Here, our main goal is to select the best compatible pair depending on the mutual attraction to ignore redundant crossover. To calculate the degree of compatibility or mutual attraction, we must calculate the female pheromones and male flash brightness value for every such pair. The detail calculation has been stated below.

Calculation of Female Pheromones

The sexual appeal of female fireflies is directly proportional to the number of pheromones released. These pheromones reach male fireflies depending on two factors: (i) wind speed, and (ii) distance between the selected pair of fireflies. Hence, the amount of pheromone for a particular female firefly reaching a corresponding male is formulated [5] as:

$$Ph_{ij} = |f_i \times dis \times w| \quad (3)$$

Here, dis is calculated by the distance between firefly i and j from the initial parent population, w is any random number within the range of 1 and 9, f_i is the calculated fitness of firefly i, and Ph_{ij} is the pheromone which is calculated for each female firefly towards every male firefly individually and an array is used to store this.

Calculation of Brightness of Flash for Male Fireflies

The female fireflies detect the sexual appeal of male fireflies towards themselves by the brightness of the flash. The brightness is directly proportional to the appeal for mating. Thus, the amount of brightness is formulated [5] as:

$$\beta_{ij} = |f_j e^{-\gamma r^2}| \quad (4)$$

Here, r is the distance between the female and male fireflies corresponding to the initial parent population, f_j is the fitness of male j, and γ is a constant between [0, 1]. The process is repeated for each male firefly towards every female firefly until we get the value for all the pairs.

Mutual Attraction

The Mutual Attraction (MuA) of each female (or male) firefly towards each male (or female) firefly is calculated by the summation of the values of the pheromone and brightness of each pair, as shown below [2].

$$MuA = \sum_{ij=0}^{MP} (Ph_{ij} + \beta_{ij}) \quad (5)$$

To store the value of mutual mating attraction of the firefly pair, we maintain a selection table, as shown in Figure 3. It contains the value of mutual attraction in the first column, female firefly identity in the second column, and male firefly identity in the third column, and the whole table is sorted according to mutual attraction. Thus, the pair having high mutual attraction among the others is considered as the first mating pair and so on.

MuA	Female	Male
.	.	.
.	.	.
.	.	.

Figure 2. Selection table storing Mutual Attraction (MuA) of considered firefly mating pair.

Mating Capability (MCap)

For each firefly, whether male or female the mating capability is dependent on the fitness computed. The mating capability determines the number of times each firefly can mate with their partners to produce new offspring which will be stored in an array respectively for both male and female. Thus, the equation to compute mating compatibility is stated below:

$$MCap = (fit / 100) * rand [1, population\_size-1] \quad (6)$$

Here, fit is the fitness of the firefly, and rand is any random value in the given range.

Crossover

In the crossover phase, firefly pairs are selected for crossover from top of the selection table pointed by table_top as mentioned in Figure 3, and for each selected male and female pair we check for the mating capability. If the capability value for both male and female firefly undergoing crossover is greater than zero, then for each firefly we set a crossover point after consecutive three cities and alternately assign selected portions to form new offspring with respect to the assigned crossover point. This process is termed as the fixed two-point crossover method. After the completion of mating, the mating compatibility is reduced by 1 for both the parents. Again, for the next round, we select the next pair of fireflies, and the whole process of crossover is carried out for m/2 times.

Mutation and Selection for the Next Generation

After crossover, the next phase is neighbourhood-based swap mutation where for each city we generate a random probability and if this probability is less than the implicit mutation probability, then for the present city, we choose randomly two locations and check whether these are permissible locations or not for swapping, i.e., if the generated location is equal to the clue index, then the pair is non-permissible. After generating a permissible pair of locations, we check for repetition in the neighbour elements, and if any repetition exists in a row or in a column, the swap operation is performed between the two locations. This process is carried out for all the subgrids. Note that in each city at most one swap is to be executed. Lastly, the mutated child is accepted only if the fitness value is greater than the fitness before mutation.

After the successful formation of offspring, we combine both the offspring and parent population and sort the whole combined two-dimensional array with respect to the fitness value and select

the alternate chromosomes for the next generation. This whole process is repeated until the global optimum of the fitness function is accomplished, or the maximum number of generations is reached.

IV. RESULTS AND DISCUSSION

The results of our proposed method are depicted in Tables 2 to judge the effectiveness and feasibility of the developed algorithm for some benchmark instances. Our algorithm is implemented on Intel® Core (TM) i5-7200U CPU @2.50GHz, 8.00GB RAM using 64-bit OS. The platform used for python coding is Anaconda 3.

Here, 50 benchmark instances for each difficulty level (based on number of cities) have been considered with epoch 10 (i.e., out of 10 runs) and population size also same as 10. The average execution time and generations for each of the difficulty levels have been included in Tables 2 and 3.

Table 1. Experimental Results based on Execution Time for Assumed Instances of Our Proposed Algorithm.

Execution Time (sec)	Easy	Medium	Hard
Minimum	1.10	1.15	4.32
Average	7.80	11.50	29.80
Maximum	20.05	53.50	152.60

V. CONCLUSION

In this paper, we have described the approach and discussed in detail the hybrid version of our proposed algorithm Hybrid Genetic Algorithm based Firefly Mating Algorithm for solving TSP. In this approach, the selection of mating pair for crossover method has been refined according to the mating structure of firefly. The constraint we assumed about the population size, which is always even and hence, the number of female fireflies is equal to the number of male fireflies, in general. The average number of generations and the execution time got reduced compared to the initial version of the algorithm.

The future work for this algorithm is to overcome the hurdles faced so far and execute this algorithm for other higher difficulty levels of TSP instances. Other works are to reduce the number of generations for the solution which could be done by improving the approach applied here. The HGFMA can be improved so that it works better for easy instances too. In this work, we have essentially concentrated in solving TSP with 9 cities, but in future, we would like to work on larger instances and to incorporate these algorithms on any TSP application.

REFERENCES

[1] T. Yato and T. Seta, “Complexity and Completeness of Finding Another Solution and Its

Application to Puzzles," IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences, vol. 86, no. 5, pp. 1052-1060, 2003.

[2] S. N. Kumbharana1, G. M. Pandey, "Solving Travelling Salesman Problem using Firefly Algorithm," International Journal for Research in Science & Advanced Technologies (IJRSAT), vol. 2, no. 2, pp. 053-057, 2013.

[3] D. Weyland, "A Critical Analysis of the Harmony Search Algorithm—How Not to Solve Sudoku," Operations Research Perspectives, vol. 2, pp. 97-105, 2015.

[4] A. Ritthipakdee, A. Thammano, N. Premasathian, and D. Jitkongchuen. "Firefly Mating Algorithm for Continuous Optimization Problems," Computational Intelligence and Neuroscience, 2017.

[5] Gerard Reinelt. The Traveling Salesman: Computational Solutions for TSP Applications. Springer-Verlag, 1994.

[6] . B. Fogel, "An Evolutionary Approach to the Traveling Salesman Problem", Biol. Cybern. 60,139144 (1988).

[7]Kylie Bryant ,Arthur Benjamin, Advisor, "Genetic Algorithms and the Traveling Salesman Problem", Department of Mathematics, December 2000.