CSC 316 2.0

A Mini report on Solving TSP by Firefly Algorithm

Introduction

Nature has been a great and immense source of inspiration for solving hard and complex problems for years. It always finds the optimal solution to problem maintaining perfect balance among its components. Nature has inspired many heuristic algorithms to obtain reasonable solutions to complex problems. Nature inspired Algorithm has been applied to a broad range of hard combinational problems, like Travelling Sales man Problem (TSP), Knapsack Problem, Graph Coloring, Vehicle Routing etc. Among them, we have selected the classic Travelling Salesman Problem (TSP).

Here we discuss and elaborate on the implementation of a program solving TSP using a new heuristic approach Firefly algorithm.

Traveling Salesman Problem (TSP)

The goal of the travelling salesman problem (TSP) is to find a tour of a given number of cities, visiting each city exactly once and returning to the starting city where the length of the tour is minimized. The TSP is a NP-hard problem, so unless we settle for an approximated result, computations will be very time consuming]. Currently the only known method guaranteed to optimally solve the travelling sales man problem of any size, is by enumerating each possible tour and searching for the tour with smallest cost. Each possible tour is a permutation of 123 . . . n, where n is the number of cities, so therefore the number of tours is n! When n gets large, it becomes impossible to find the cost of every tour in polynomial time. Such a method, which will end up giving the optimal solution, is obviously not very feasible because of computing power and time. It can be seen that even for small instances, the time consumption is extremely high if we want to find every possible tour. Hence we use an approximation algorithm, which in less time will end up giving a result that isn't necessarily the best tour, but instead a tour that is close to the best tour.

Firefly algorithm

Firefly Algorithm, as a random optimization algorithm, has simulated the luminescent behavior of fireflies to search for food in Nature, putting the search and optimization process of fireflies as the process of individual firefly attraction and location update. In the algorithmic optimization process, fluorescent brightness and fireflies attraction are two key factors in the evolution of firefly locations. Fluorescent brightness is related to the location of individual firefly as the better location, the higher brightness. The individual firefly with a bad location would move in a direction that depends on the firefly attraction with a good location as the larger attraction of the firefly, the easier it is to attract other fireflies closing to its location. When two types of fluorescent brightness are the same, the individual firefly will be randomly oscillating. With the constant update on individual firefly fluorescent brightness and attraction, firefly swarm will eventually come together and achieve the algorithmic optimization effect

PSEUDO CODE for firefly algorithm
irefly Algorithm
Objective function $f(x)$, $x = (x_1, \ldots, x_d)^T$
Generate initial population of fireflies xi ($i = 1, 2, \ldots, n$)
Light intensity I_i *at* x_i
is determined by $f(x_i)$
Define light absorption coefficient γ
while ($t <$ *MaxGeneration*)
for $i = 1$: n *all n fireflies*
for $j = 1$: i *all n fireflies*
if ($I_j > I_i$), *Move firefly i towards j in d – dimension*; *end if*
Attractiveness varies with distance r via exp$[-r]$
Evaluate new solutions and update light intensity
end for j
end for i
Rank the fireflies and find the current best
end while
Postprocess results Rank the fireflies and find the current best;
End while;
Post process results and visualization;
End procedure

Code Implementation and Approach

I have used the programming language c sharp to implement the firefly algorithm for tsp. As its easier to create an interface in Visual Studio and better to implement Firefly algorithm in OOP.

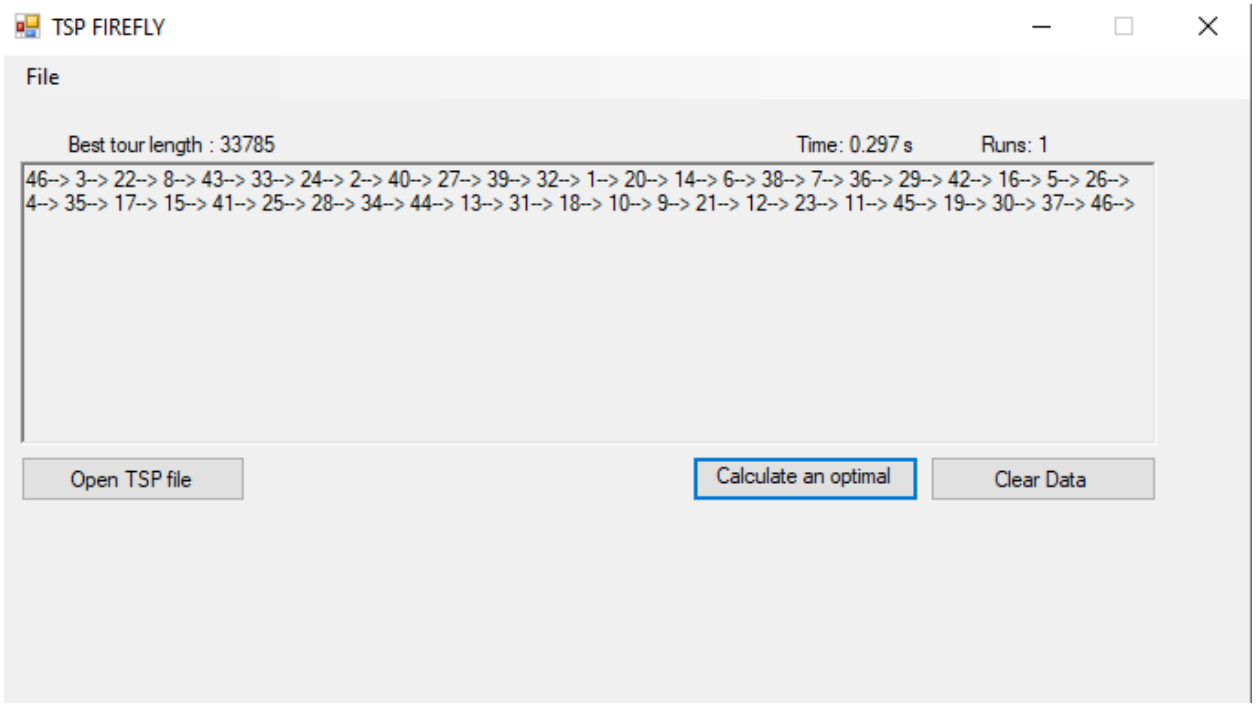
The classes “City”, ”Fireflies” and “Heuristic” were implemented with relevant functions.
Function to calculate the distance between two cities is implemented within the City class.
TSP file is read and coordinates are used to create City objects(TSPFirefly class).

After clicking **Calculate the optimal,** the list of the cities is passed to function *start* through a new Fireflies object. To initialize a solution Nearestneighbourhood method is implemented in Heuristic class and is called in *start* function.

Each step in firefly algorithm is implemented as separate functions and called within the start function. the The number of iterations is fixed to 500, increasing this number have crashed the application. The best solution of the list of cities in the order of the tour is returned. The city at which the tour started is again added to the last of the tour, to show the loop of the tour.
And the distance of the best tour is calculated in TSPFirefly. And also Elapsed time is recorded and shown in the application as well. Moreover the path of the best tour is also shown in the interface.
We can also run this algorithm again with the current best route solution given as the input for firefly algorithm. The number of runs is also counted and displayed.

Clear data button is used to call the function to clear all the set parameters and start with a new TSP file.

5 TSP files are downloaded from TSPLIB and were input to this program.



Results

Best tour is calculated in one run each (500 iterations)

City TSP	# of cities	Best tour length	Elapsed time
att48	48	33785	0.297seconds
eli101	101	675	0.499 seconds
gr17	17	2026	0.194 seconds
xql662	662	3091	2.534 seconds
pr226	226	86876	0.836 seconds

Table 1

Best tour vs runs is tabulated for the above TSP cities

Runs	Best tour for gr17	Best tour for pr226	Best tour for att48	Best tour for eli101	Best tour for xql662
1	2026	84568	34534	698	3129
2	2026	84178	33495	686	3086
3	2026	83221	32952	679	3059
4	2026	82876	32911	678	3058
5	2026	82876	32911	664	3058
6	2026	82876	32786	660	3045
7	2026	82839	32786	659	3037
8	2026	82412	32786	659	3036
9	2026	82289	32786	659	3032
10	2026	82289	32786	659	3015

`

Table 2

For different Population size, the best tour and elapsed time are plotted for pr226

Populations size	Best tour	Elapased time
10	87772	0.893
20	86708	1.8
30	88277	1.178
40	87764	1.415
50	88182	1.755
60	88658	1.764
70	86101	2.28
80	90203	2.258

*note : the population is changed manually (which was 10 default) in the codes and was rebuilt at each population size

Discussions and Interpretations

In table 1 we can see that the elapsed time increases with the number of cities. And in table 2 we can identify that after each run a more optimal solution is found and for gr17 and att48, the best tour length, is repeating after a certain number of runs which may be considered as the best solution. In table 3 we can show that elapsed time is increasing with the population size but cannot predict a relationship between the population size and the best route length.

Conclusion

A Firefly algorithm to solve Traveling Salesman Problem is presented in this code implementation. Firefly Algorithm produces very good results for all types, small to big TSP instances in comparison with the results produced by the other algorithms. It is also observed that the algorithm starts to take more time to converge into the solution with the increase in its

size. This algorithm produces very good results and can be applied to solve any kind of optimization problem.