University of Malta

Ant-Colony-Optimization to solve instances of the TSP

Classification, Search and Optimisation: Assignment

Author
Matthew Vella (428698M)

04th Janaury 2020

Contents

1 Introduction

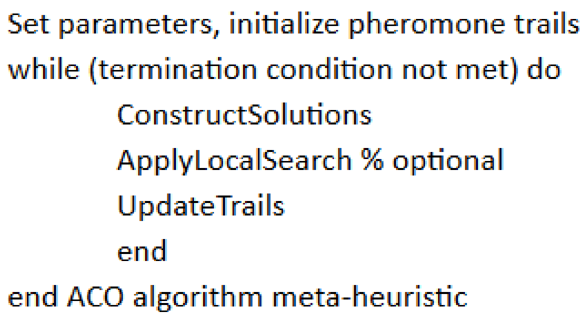
Traveling salesman problem (TSP) belongs to the class of optimization problems. The problem can be formulated as determining the shortest route of minimal length that passes through each city exactly one time. This task belongs to the class of NP problems. It is very easy to describe such kind of problem and very difficult to solve it. Therefore, scientists have been developing efficient algorithms since 1950's. In terms of graph theory, TSP can be formulated as finding the Hamiltonian circuit with the least weight for a given weighted complete graph. The TSP has numerous applications in various fields of science and engineering [1]. The naive algorithm to solve SP has clearly non-polynomial complexity which is very inefficient. Instead, a lot of approximate methods are used, for instance, genetic algorithms (GA) [2], simulated annealing [3], tabu search [4], ant colony optimization[5-6], and neural networks (NNs) [7] that show rather good results. Also, some exact algorithms were developed that allow solving TSP problem for large input [1].

Ant Colony Optimization (ACO) is a method which can solve a number of NP problems. Originally, it was applied to TSP. ACO algorithms are used for solving optimization tasks. The applications of ACO algorithms for engineering problem solving are proposed in papers [8, 9]. ACO techniques use a conception of virtual ants that act as the agents that communicate between each others, and employs randomly propagation rules. ACO methods possess many crucial parameters that greatly affects the running time and they can not be easily determined. In the present study, ACO algorithms were used to solving TSP problem for a number of instances.

2 Literature Review: How ACO works and how is this applicable to TSP

2.1 Ant Colony Optimization

Ant Colony Optimization was introduced by Dorigo et al. in [5] and were further developed in [10-12].
In ACO, artificial ants construct solutions to the given optimization problem and provide information on the solutions quality via the communication channel that resembles those of real ants [13]. The scheme of ACO algorithm is the following:



```
Set parameters, initialize pheromone trails
while (termination condition not met) do
        ConstructSolutions
        ApplyLocalSearch % optional
        UpdateTrails
        end
end ACO algorithm meta-heuristic
```

It was indicated earlier, that after the parameters and pheromone trail were initialized, every ant proposes its own solution in a probabilistic manner and updates pheromone level on the graph respectively. Also, performance can be improved by applying local search methods to particular solutions. The algorithm is described in details below:

1. ConstructSolutions: Every ant proposes a state, and walks through the states in a sequential manner. At every state, every ant determines probability density function to select one of the states according to this distribution. This random selection is known as random proportional transition rule at is affected by two parameters: Move Attractiveness. It is a function that indicates the priory move attractiveness. The pheromone level on the trail. It expresses the posteriori move attractiveness since it shows the learned attractiveness of selecting the further possible states while being in the current state. Due to that, every ant build a solution step by step in order to solve the problem for a certain input.

2. ApplyLocalSearch: Before updating trail level, some local search methods can be used for each current iteration solution. This step is not mandatory. However, it can improve the obtained solutions and is employed in many ACO implementations.

3. UpdateTrails: When the solutions were build and evaluated, pheromone level on the trails changes depending on the quality of solutions, respectively.

The ACO methods that were used in this study, are: ant colony system (ACS) [14], max–min ant system (MMAS) [15], and elitist ant system (ASe) [16]. A ACO algorithms differs in the part of pheromone updating and random proportional transition rule which are the pillars of the corresponding algorithm.

2.2 Random proportional transition rule

In the ant system for TSP, the transition probability from node i to node j can be determined by the following formula:

$$p_{ij} = \begin{cases} \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{h\in\Omega}[\tau_{ih}]^{\alpha}[\eta_{ih}]^{\beta}} & \text{if } j \in \Omega \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where t_{ij} is trail intensity between nodes i and j, n_{ij} is reach-ability of node j from node i ($n_{ij} = 1/d_{ij}$), $n_{ij} = 1/d_{ij}$, α is the set of not visited nodes, d_{ij} is the distance between nodes i and j. The trail intensity is affected by a parameter α. The mentioned criteria is treated an attractiveness measure and is affected by a parameter β.
It is worth noting, that constants α and β affects greatly the algorithm running time. The parameters best values are often determined from the numerical experiments.
From three mentioned algorithms, only ant colony system (ACS) does not rely on the described above RPT rule. They differ only in the pheromone updating strategy. In ACS, the next approach is used:

$$j = \begin{cases} \arg \quad \max_{j\in\Omega}\left\{[\tau(i,j)] \cdot [\eta(i,j)]^{\beta}\right\} & \text{if } q \leqslant q_0 \\ J & \text{otherwise} \end{cases} \tag{2}$$

where q is a generated random number, q_0 is a parameter, $q_0[0,1]$, and J is random variable chosen according to probability distribution function defined in Eq. (1). At every step, when an agent in node i decides to traverse to node j, a random numbed q_0 is generated from uniform distribution [41].

2.3 Pheromone Updating

Pheromone updating gives two types of feedback in ants communication. Pheromone on the good solutions provides positive feedback. Nevertheless, the increase of pheromone level without reducing its quantity leads to the non-optimal state. To mitigate this, negative feedback mechanism is provided by pheromone evaporation the ants paths. In AS, at the every iteration t end, every agent k places $1/L_k$(here L_k is the route length) amount of pheromone and pheromone is evaporated from all edges by t_{ij} $(t+1) = (1-p)t_{ij}(t)$ equation where $p(p[0,1])$ is the decay coefficient.
Elitist strategy is the changed version of AS algorithm. At each cycle the trail that was put on the edges belonging to the best tour is reinforced more than in AS by the elitist ants. After every cycle, extra e/L_* level of pheromone is added to the trail of each ant of the best found tour T^*, where e is the elitist ants number and L is the length of best tour from the beginning of trail.
ACS employs two kinds of pheromone updating strategies: local and global. In global updating, after every iteration, only pheromones on best so far tour are updated by increasing pheromone level with $p/(L^{*}_{)}$ where p is the decay parameter. Local pheromone update is performed after each step of tour by all ants using the next formula:

$$\tau_{ij} = (1-\varphi) \cdot \tau_{ij} + \varphi \cdot \tau_0$$

where $p(p(0,1])$ is the local pheromone decay coefficient, t_0 is the initial pheromone value.
In Max-Min AS, pheromone is placed only on the best found tour like ACS. The pheromone level is reinitialized when a new best tour was detected. To avoid non-optimal solutions, two rules were introduced.

- The pheromone level is restricted to range $[t_{min}, t_{max}]$;
- The pheromone value is initialized as t_{max}.

3 Genetic Algorithms VS ACO to solve the TSP

Genetic algorithms belong to evolutionary computing algorithms. Genetic algorithms use ideas of Darwin's evolution theory. The solution to a problem solved by genetic algorithms is evolved in time. I.Rechenberg introduced the idea of evolutionary computing in the 1960s [17]. Other researchers then developed his idea. Genetic Algorithms (GAs) were invented by John Holland and further worked out in [18].

3.1 Basic Description of GA

Genetic algorithm is started with a set of solutions called population. Each solution is represented by chromosomes. Solutions from one population are taken and used to form a new population. One hopes that the new population will be better than the old one. Solutions which are selected to form new solutions, the so called offsprings are selected according to their fitness. The more suitable they are the more chances they have to reproduce. This is repeated until some condition is met. For instance, the condition can be the number of populations or improvement of the best solution. It is known that many optimization problems can often be transformed as finding the extreme of a function. This is true for the following problem: for the give function GA tries to find its optimum.

3.2 The main steps of genetic algorithms are described below [19].

- STEP0 Q=generateinitialpopulation( ) // Initialize population Q
- STEP1 F=calculateobjectfitness(Q) // calculated fitness(F) of population Q
- STEP2 FOR i=1 to T BEGIN // T is the iteration step
- STEP3 selectoperator(Q,F) // Selection operator
- STEP4 crossoveroperator(Q,pc) // crossover operator
- STEP5 mutationoperator(Q,F,pm) // mutation operator
- STEP6 F=calculateobjectfitness(Q) // calculated fitness(F) of population Q
- STEP7 END

The aforementioned GA algorithm is very general. There can be some alterations depending on the problem.
The first question is regarding chromosomes creation and appropriate encoding. The next question is select parents selection. Creating a new population only by new children can lead to loss of the best chromosome from the previous population. This is true, so a method called Elitism is often used. So at least one best solution is taken as is a new population, so this solution can be kept to the end of the run.
The crossover and mutation are the most fundamental steps of GA, that affects its performance. The main parameters of GA are: Crossover Probability, Mutation Probability and Population size. Crossover probability determines the frequency of crossover. Without crossover, offspring is identical to the parents. In the case of crossover, offspring consists of parent's chromosome parts. If crossover probability is equal to 1 then all offspring are obtained as a result of crossover. If it is 0, the entire new generation consists from old population exact copies of chromosomes. This does not necessarily mean that the new generation is the same as previous. Crossover is performed hoping that new chromosomes will have better parts than old chromosomes. However, it is advisable to take a portion of the population to the next generation.
Mutation probability defines the frequency of a chunk of chromosome mutation. Without mutation, the child is taken after crossover intact. In case of mutation, a portion of chromosome alters. If mutation probability is equal to 1, the entire chromosome alters. In case this quantity equals to 0, everything is taken as is. Mutation is made to avoid local extremes, but it should not happen too often, since in this case GA eventually change to random search. Population size is the number of chromosomes in one generation of population. If there is only a couple of chromosomes, GA has a small number of crossover possibilities. In this case a small area of search space is studied. On the other hand, if there are too many chromosomes, GA slows down. According to the research results, after some population size it is not useful to increase it, since one does not get the decrease of computational time. This size mainly depends on encoding and the problem itself.
Chromosomes are chosen from the population to be parents to crossover. The problem is how to select these chromosomes. According to Darwin's theory the best ones should survive and create new offspring. The main techniques to choose the best chromosomes are as follows: Boltzman selection, Tournament selection, Rank selection, Roulette wheel selection, Steady state selection. Encoding of chromosomes is coupled with GA method. Encoding depends greatly on the problem; there are many encoding style like: Binary encoding, Permutation encoding, Value encoding, or Tree encoding. First, we need to

choose how to represent the route. Path representation is the most evident way to encode the route. Each city can be assigned an alphanumeric name. In this case, the route is a chromosome. New routes are being created by means of certain genetic operators. For TSP, one often uses the Permutation encoding to order the problem. Every chromosome is a string of numbers, which represents the numbers in a sequence. Permutation encoding is only useful for ordering problems. For such type of problems and certain variants of crossover and mutation some changes must performed to leave the chromosome intact, so that it encodes really existing sequence. TSP is a good example of permutation encoding. Chromosomes shows the order of cities, in which salesman will visit them.
The crossover and mutation operators are largely determined by encoding type and the problem itself. For TSP encoding and problem, one often uses single point crossover. The permutation is copied from the first parent until we reach this point, then the second parent is scanned and if the number is not yet in the offspring it is added. There exists a number of ways to get the chromosome part after the crossover point.

3.3 Encoding and Decoding

The decimal coding can be used to encode the route. For instance, a chromosome 138,246,795 represent a path from the node, followed by the nodes 3,8,2,4,6,7,9,5 and finally to the node 1. Fitness function Fitness function is inversely proportional to the route distance: $f = 1/s$, here $s =_(i = 1)^N d_(i, i+1)$ is the path length. Selection operator Classic roulette wheel method is mostly used for operator selection. First we calculate the fitness of each individual. Afterwards, the probability of individual selection is determined. The next generation of individual is selected by roulette wheel method.

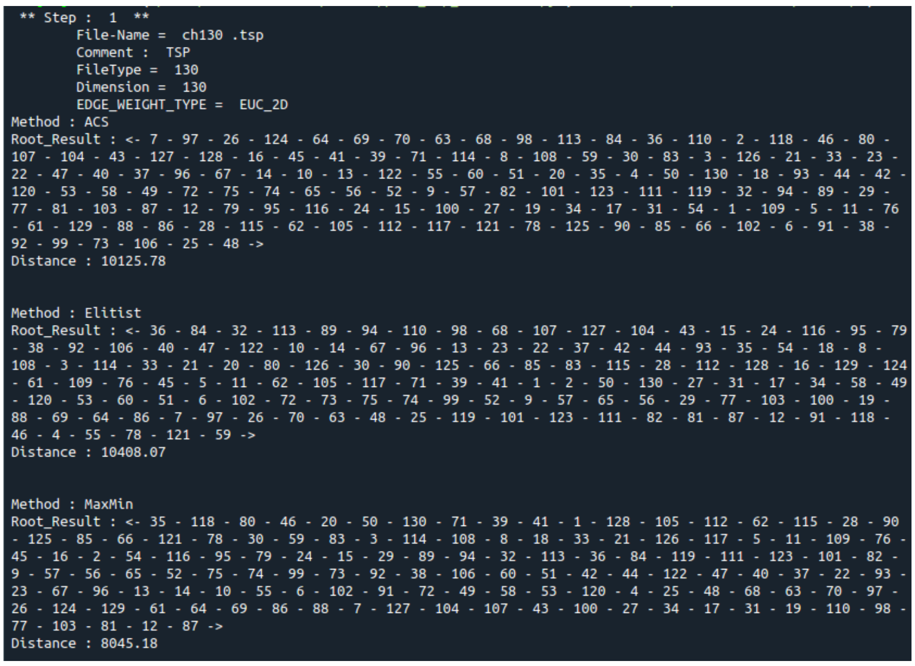
4 Our Solution using ACO to solve the TSP

4.1 Description of our Implementation

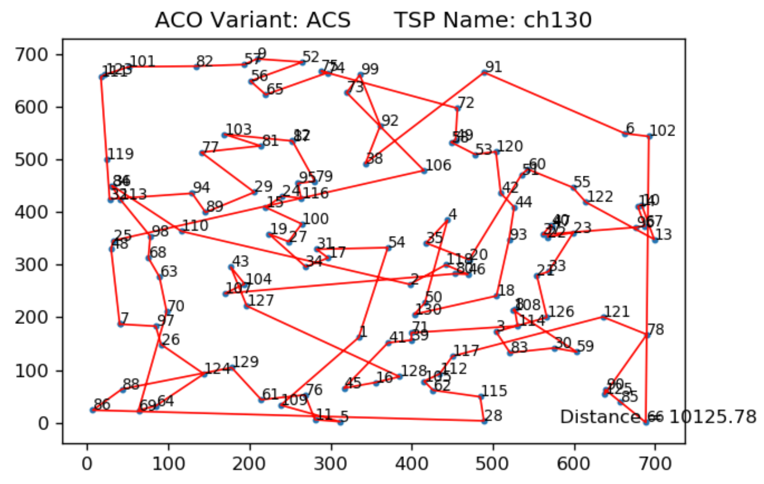
In current study, three ACO methods were implemented. They are Ant Colony system (ACS), Elitist Ant System and Max-Min Ant System. In each of these methods, colony size was equal to 5. Number of iterations was 50. Parameters α=1 and β=3. Parameter $p = 0.1$. Roulette wheel selection (RWS) strategy was used. In this method, every solution takes a segment in a wheel proportional to its fitness. Each time an individual is to be chosen, a random number is generated for the roulette wheel position. The individual is chosen with respect to the determined segment of the roulette wheel.

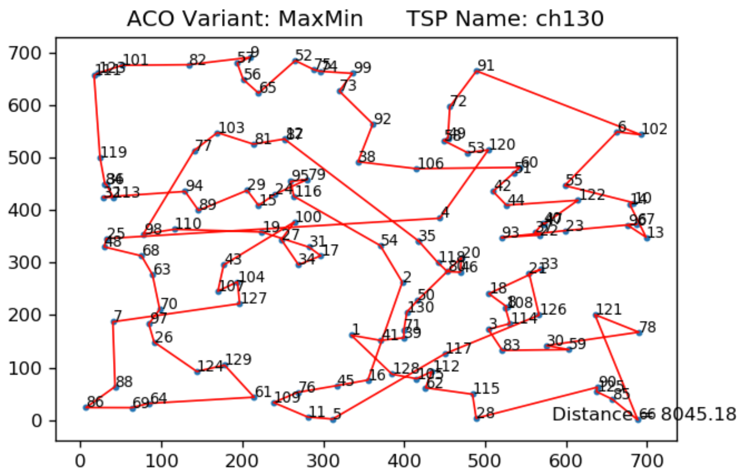
4.2 Results

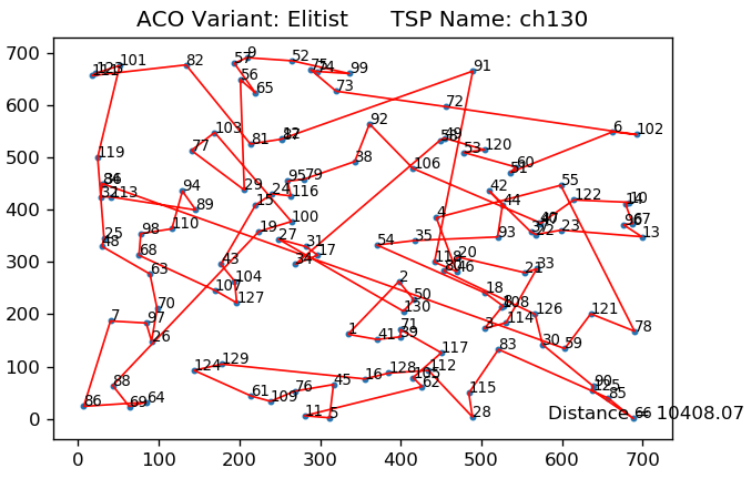
ch130.tsp



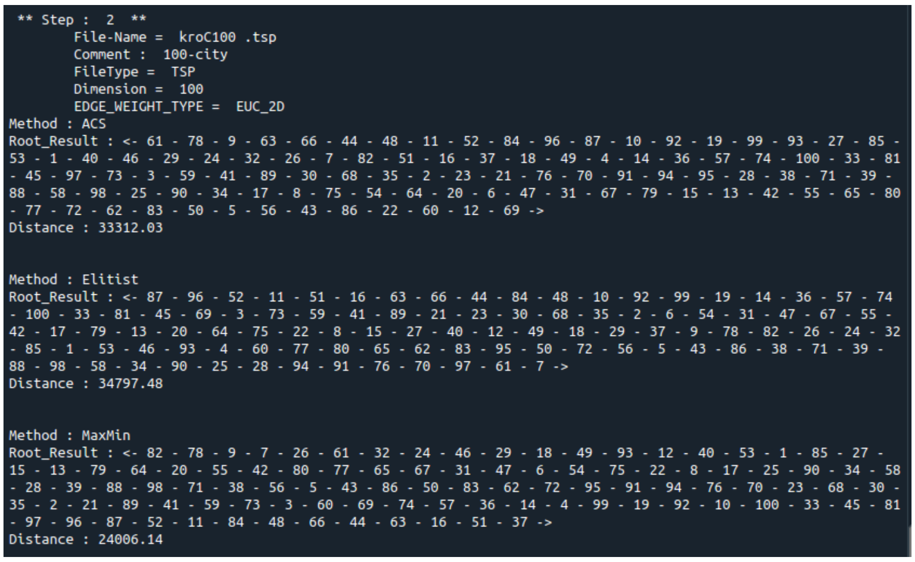
```
 ** Step :  1  **
        File-Name =  ch130 .tsp
        Comment :  TSP
        FileType =  130
        Dimension =  130
        EDGE_WEIGHT_TYPE =  EUC_2D
Method : ACS
Root_Result : <- 7 - 97 - 26 - 124 - 64 - 69 - 70 - 63 - 68 - 98 - 113 - 84 - 36 - 110 - 2 - 118 - 46 - 80 -
107 - 104 - 43 - 127 - 128 - 16 - 45 - 41 - 39 - 71 - 114 - 8 - 108 - 59 - 30 - 83 - 3 - 126 - 21 - 33 - 23 -
22 - 47 - 40 - 37 - 96 - 67 - 14 - 10 - 13 - 122 - 55 - 60 - 51 - 20 - 35 - 4 - 50 - 130 - 18 - 93 - 44 - 42 -
120 - 53 - 58 - 49 - 72 - 75 - 74 - 65 - 56 - 52 - 9 - 57 - 82 - 101 - 123 - 111 - 119 - 32 - 94 - 89 - 29 -
77 - 81 - 103 - 87 - 12 - 79 - 95 - 116 - 24 - 15 - 100 - 27 - 19 - 34 - 17 - 31 - 54 - 1 - 109 - 5 - 11 - 76
- 61 - 129 - 88 - 86 - 28 - 115 - 62 - 105 - 112 - 117 - 121 - 78 - 125 - 90 - 85 - 66 - 102 - 6 - 91 - 38 -
92 - 99 - 73 - 106 - 25 - 48 ->
Distance : 10125.78


Method : Elitist
Root_Result : <- 36 - 84 - 32 - 113 - 89 - 94 - 110 - 98 - 68 - 107 - 127 - 104 - 43 - 15 - 24 - 116 - 95 - 79
- 38 - 92 - 106 - 40 - 47 - 122 - 10 - 14 - 67 - 96 - 13 - 23 - 22 - 37 - 42 - 44 - 93 - 35 - 54 - 18 - 8 -
108 - 3 - 114 - 33 - 21 - 20 - 80 - 126 - 30 - 90 - 125 - 66 - 85 - 83 - 115 - 28 - 112 - 128 - 16 - 129 - 124
- 61 - 109 - 76 - 45 - 5 - 11 - 62 - 105 - 117 - 71 - 39 - 41 - 1 - 2 - 50 - 130 - 27 - 31 - 17 - 34 - 58 - 49
- 120 - 53 - 60 - 51 - 6 - 102 - 72 - 73 - 75 - 74 - 99 - 52 - 9 - 57 - 65 - 56 - 29 - 77 - 103 - 100 - 19 -
88 - 69 - 64 - 86 - 7 - 97 - 26 - 70 - 63 - 48 - 25 - 119 - 101 - 123 - 111 - 82 - 81 - 87 - 12 - 91 - 118 -
46 - 4 - 55 - 78 - 121 - 59 ->
Distance : 10408.07


Method : MaxMin
Root_Result : <- 35 - 118 - 80 - 46 - 20 - 50 - 130 - 71 - 39 - 41 - 1 - 128 - 105 - 112 - 62 - 115 - 28 - 90
- 125 - 85 - 66 - 121 - 78 - 30 - 59 - 83 - 3 - 114 - 108 - 8 - 18 - 33 - 21 - 126 - 117 - 5 - 11 - 109 - 76 -
45 - 16 - 2 - 54 - 116 - 95 - 79 - 24 - 15 - 29 - 89 - 94 - 32 - 113 - 36 - 84 - 119 - 111 - 123 - 101 - 82 -
9 - 57 - 56 - 65 - 52 - 75 - 74 - 99 - 73 - 92 - 38 - 106 - 60 - 51 - 42 - 44 - 122 - 47 - 40 - 37 - 22 - 93 -
23 - 67 - 96 - 13 - 14 - 10 - 55 - 6 - 102 - 91 - 72 - 49 - 58 - 53 - 120 - 4 - 25 - 48 - 68 - 63 - 70 - 97 -
26 - 124 - 129 - 61 - 64 - 69 - 86 - 88 - 7 - 127 - 104 - 107 - 43 - 100 - 27 - 34 - 17 - 31 - 19 - 110 - 98 -
77 - 103 - 81 - 12 - 87 ->
Distance : 8045.18
```



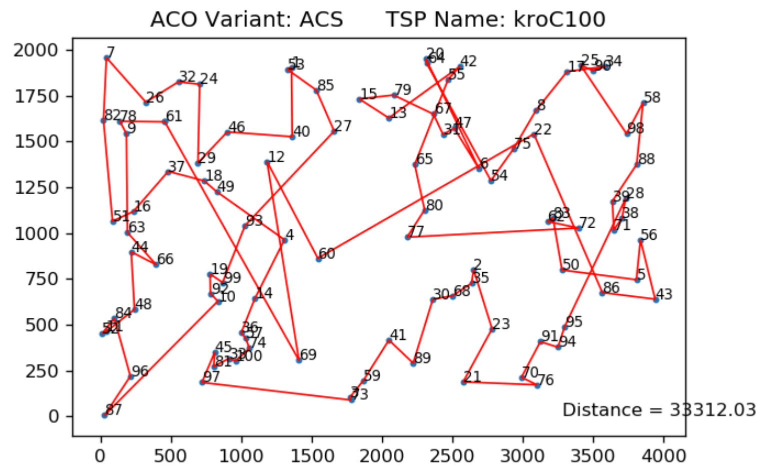

ACO Variant: MaxMin TSP Name: ch130
Distance 8045.18

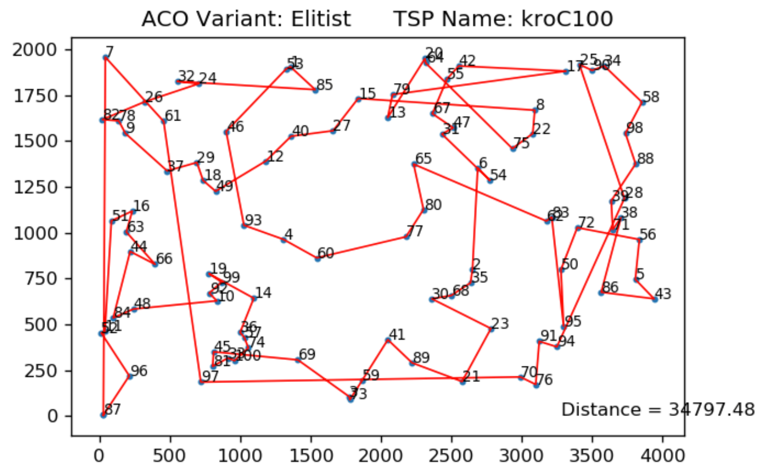

ACO Variant: Elitist TSP Name: ch130
Distance 10408.07

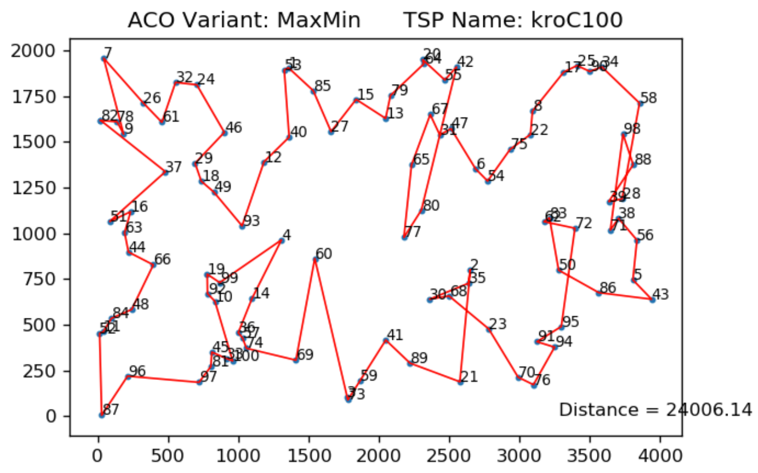
kroC100.tsp



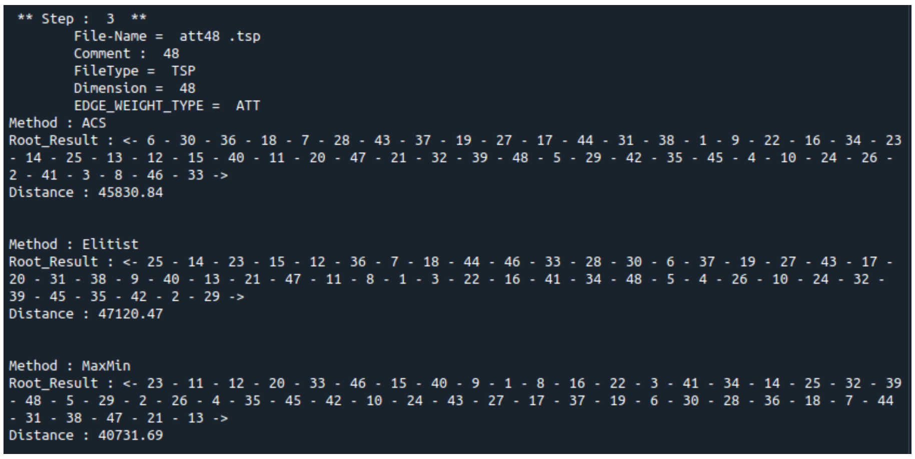
```
 ** Step :  2  **
        File-Name =  kroC100 .tsp
        Comment :  100-city
        FileType =  TSP
        Dimension =  100
        EDGE_WEIGHT_TYPE =  EUC_2D
Method : ACS
Root_Result : <- 61 - 78 - 9 - 63 - 66 - 44 - 48 - 11 - 52 - 84 - 96 - 87 - 10 - 92 - 19 - 99 - 93 - 27 - 85 -
53 - 1 - 40 - 46 - 29 - 24 - 32 - 26 - 7 - 82 - 51 - 16 - 37 - 18 - 49 - 4 - 14 - 36 - 57 - 74 - 100 - 33 - 81
- 45 - 97 - 73 - 3 - 59 - 41 - 89 - 30 - 68 - 35 - 2 - 23 - 21 - 76 - 70 - 91 - 94 - 95 - 28 - 38 - 71 - 39 -
88 - 58 - 98 - 25 - 90 - 34 - 17 - 8 - 75 - 54 - 64 - 20 - 6 - 47 - 31 - 67 - 79 - 15 - 13 - 42 - 55 - 65 - 80
- 77 - 72 - 62 - 83 - 50 - 5 - 56 - 43 - 86 - 22 - 60 - 12 - 69 ->
Distance : 33312.03


Method : Elitist
Root_Result : <- 87 - 96 - 52 - 11 - 51 - 16 - 63 - 66 - 44 - 84 - 48 - 10 - 92 - 99 - 19 - 14 - 36 - 57 - 74
- 100 - 33 - 81 - 45 - 69 - 3 - 73 - 59 - 41 - 89 - 21 - 23 - 30 - 68 - 35 - 2 - 6 - 54 - 31 - 47 - 67 - 55 -
42 - 17 - 79 - 13 - 20 - 64 - 75 - 22 - 8 - 15 - 27 - 40 - 12 - 49 - 18 - 29 - 37 - 9 - 78 - 82 - 26 - 24 - 32
- 85 - 1 - 53 - 46 - 93 - 4 - 60 - 77 - 80 - 65 - 62 - 83 - 95 - 50 - 72 - 56 - 5 - 43 - 86 - 38 - 71 - 39 -
88 - 98 - 58 - 34 - 90 - 25 - 28 - 94 - 91 - 76 - 70 - 97 - 61 - 7 ->
Distance : 34797.48


Method : MaxMin
Root_Result : <- 82 - 78 - 9 - 7 - 26 - 61 - 32 - 24 - 46 - 29 - 18 - 49 - 93 - 12 - 40 - 53 - 1 - 85 - 27 -
15 - 13 - 79 - 64 - 20 - 55 - 42 - 80 - 77 - 65 - 67 - 31 - 47 - 6 - 54 - 75 - 22 - 8 - 17 - 25 - 90 - 34 - 58
- 28 - 39 - 88 - 98 - 71 - 38 - 56 - 5 - 43 - 86 - 50 - 83 - 62 - 72 - 95 - 91 - 94 - 76 - 70 - 23 - 68 - 30 -
35 - 2 - 21 - 89 - 41 - 59 - 73 - 3 - 60 - 69 - 74 - 57 - 36 - 14 - 4 - 99 - 19 - 92 - 10 - 100 - 33 - 45 - 81
- 97 - 96 - 87 - 52 - 11 - 84 - 48 - 66 - 44 - 63 - 16 - 51 - 37 ->
Distance : 24006.14
```



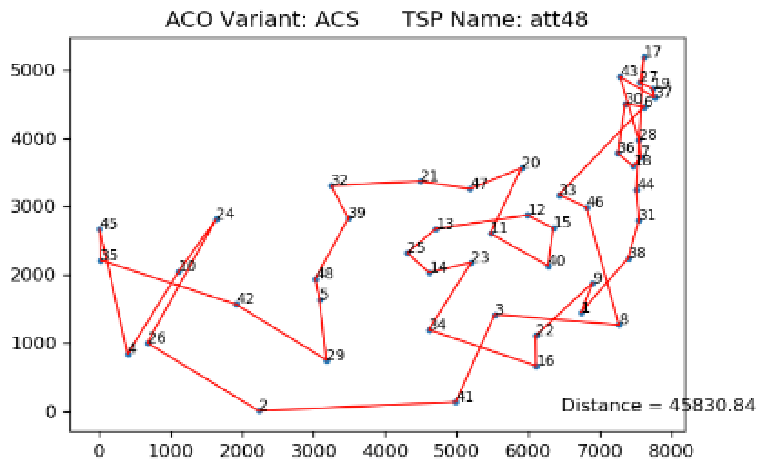


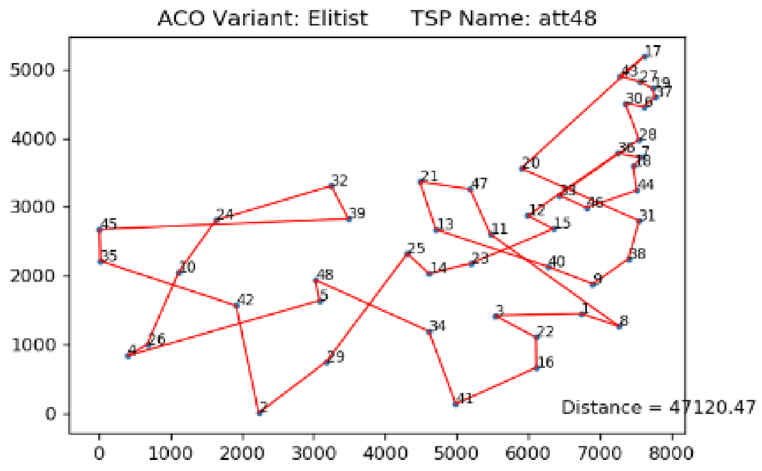


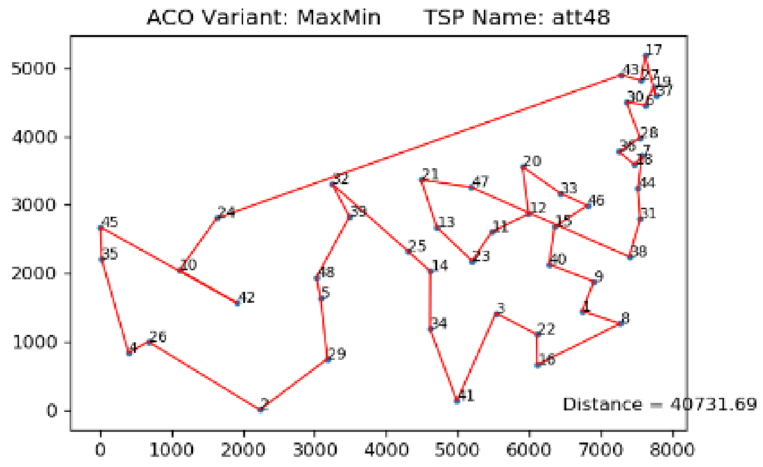
att48.tsp



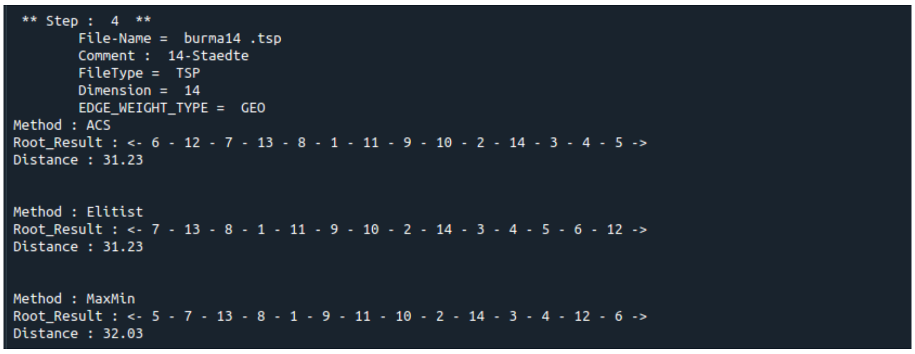
```
 ** Step :  3  **
        File-Name =  att48 .tsp
        Comment :  48
        FileType =  TSP
        Dimension =  48
        EDGE_WEIGHT_TYPE =  ATT
Method : ACS
Root_Result : <- 6 - 30 - 36 - 18 - 7 - 28 - 43 - 37 - 19 - 27 - 17 - 44 - 31 - 38 - 1 - 9 - 22 - 16 - 34 - 23
- 14 - 25 - 13 - 12 - 15 - 40 - 11 - 20 - 47 - 21 - 32 - 39 - 48 - 5 - 29 - 42 - 35 - 45 - 4 - 10 - 24 - 26 -
2 - 41 - 3 - 8 - 46 - 33 ->
Distance : 45830.84


Method : Elitist
Root_Result : <- 25 - 14 - 23 - 15 - 12 - 36 - 7 - 18 - 44 - 46 - 33 - 28 - 30 - 6 - 37 - 19 - 27 - 43 - 17 -
20 - 31 - 38 - 9 - 40 - 13 - 21 - 47 - 11 - 8 - 1 - 3 - 22 - 16 - 41 - 34 - 48 - 5 - 4 - 26 - 10 - 24 - 32 -
39 - 45 - 35 - 42 - 2 - 29 ->
Distance : 47120.47


Method : MaxMin
Root_Result : <- 23 - 11 - 12 - 20 - 33 - 46 - 15 - 40 - 9 - 1 - 8 - 16 - 22 - 3 - 41 - 34 - 14 - 25 - 32 - 39
- 48 - 5 - 29 - 2 - 26 - 4 - 35 - 45 - 42 - 10 - 24 - 43 - 27 - 17 - 37 - 19 - 6 - 30 - 28 - 36 - 18 - 7 - 44
- 31 - 38 - 47 - 21 - 13 ->
Distance : 40731.69
```



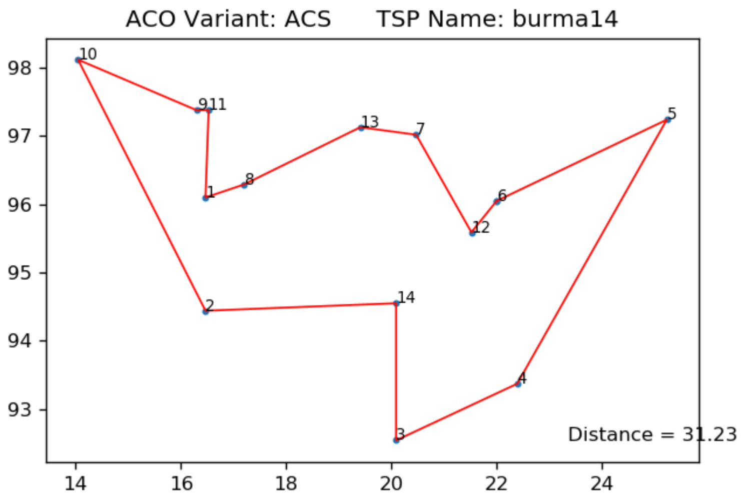


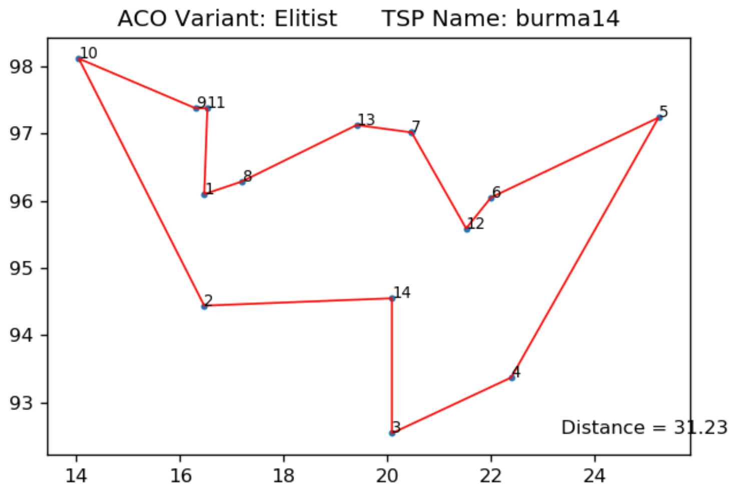


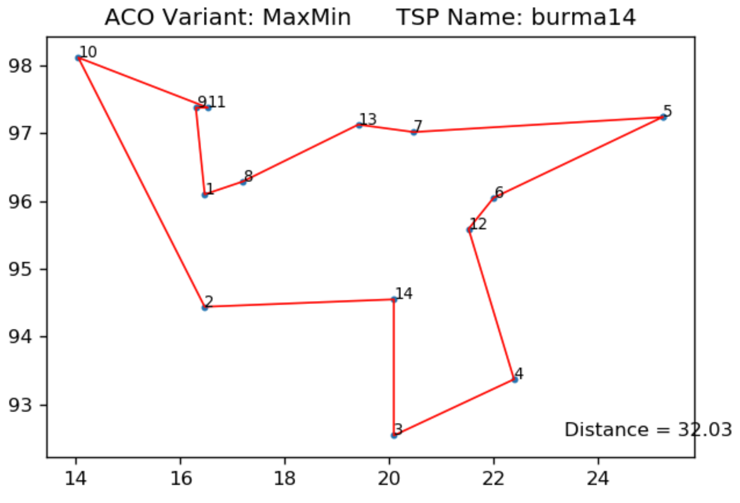
burma14.tsp



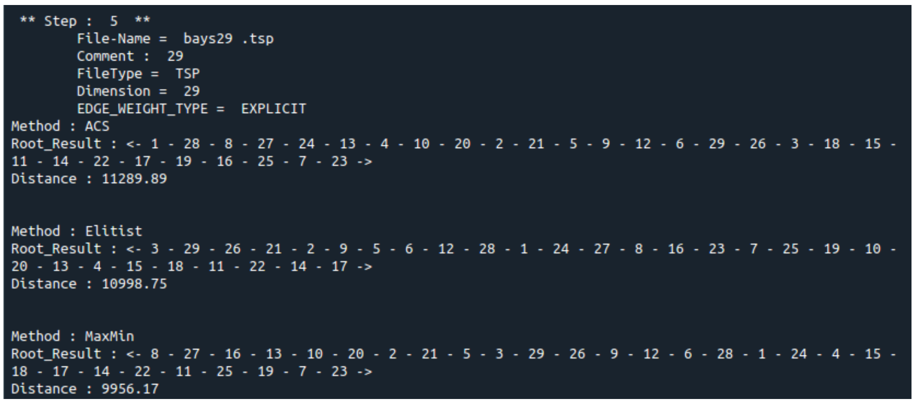
```
 ** Step :  4  **
        File-Name =  burma14 .tsp
        Comment :  14-Staedte
        FileType =  TSP
        Dimension =  14
        EDGE_WEIGHT_TYPE =  GEO
Method : ACS
Root_Result : <- 6 - 12 - 7 - 13 - 8 - 1 - 11 - 9 - 10 - 2 - 14 - 3 - 4 - 5 ->
Distance : 31.23


Method : Elitist
Root_Result : <- 7 - 13 - 8 - 1 - 11 - 9 - 10 - 2 - 14 - 3 - 4 - 5 - 6 - 12 ->
Distance : 31.23


Method : MaxMin
Root_Result : <- 5 - 7 - 13 - 8 - 1 - 9 - 11 - 10 - 2 - 14 - 3 - 4 - 12 - 6 ->
Distance : 32.03
```


ACO Variant: ACS TSP Name: burma14
Distance = 31.23

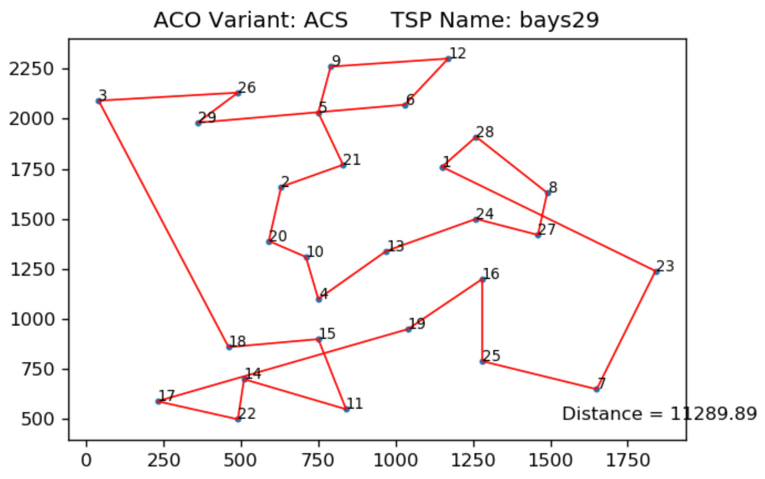

ACO Variant: Elitist TSP Name: burma14
Distance = 31.23

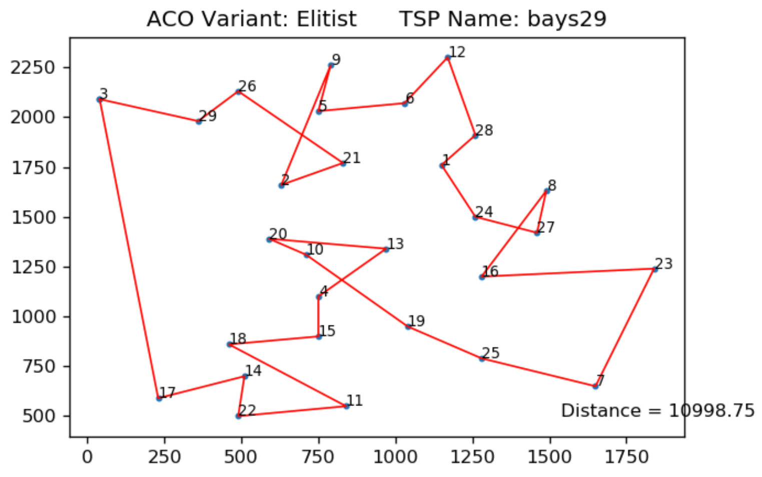

ACO Variant: MaxMin TSP Name: burma14
Distance = 32.03

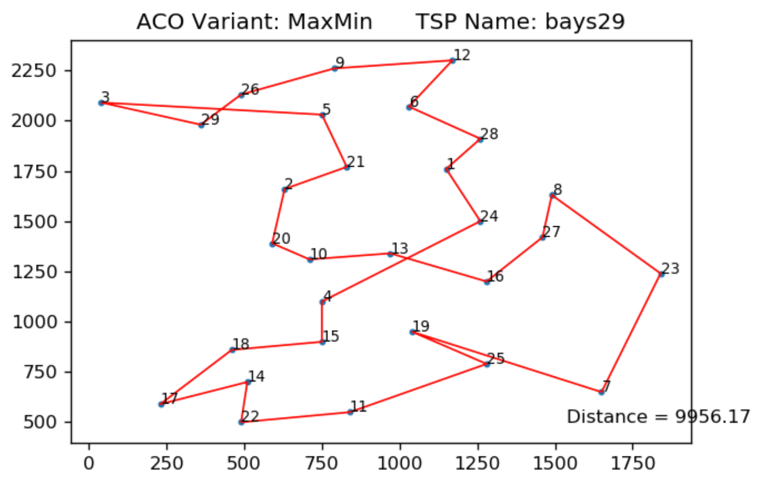
bays29.tsp



```
 ** Step :  5  **
        File-Name =  bays29 .tsp
        Comment :  29
        FileType =  TSP
        Dimension =  29
        EDGE_WEIGHT_TYPE =  EXPLICIT
Method : ACS
Root_Result : <- 1 - 28 - 8 - 27 - 24 - 13 - 4 - 10 - 20 - 2 - 21 - 5 - 9 - 12 - 6 - 29 - 26 - 3 - 18 - 15 -
11 - 14 - 22 - 17 - 19 - 16 - 25 - 7 - 23 ->
Distance : 11289.89


Method : Elitist
Root_Result : <- 3 - 29 - 26 - 21 - 2 - 9 - 5 - 6 - 12 - 28 - 1 - 24 - 27 - 8 - 16 - 23 - 7 - 25 - 19 - 10 -
20 - 13 - 4 - 15 - 18 - 11 - 22 - 14 - 17 ->
Distance : 10998.75


Method : MaxMin
Root_Result : <- 8 - 27 - 16 - 13 - 10 - 20 - 2 - 21 - 5 - 3 - 29 - 26 - 9 - 12 - 6 - 28 - 1 - 24 - 4 - 15 -
18 - 17 - 14 - 22 - 11 - 25 - 19 - 7 - 23 ->
Distance : 9956.17
```







4.3 Evaluation

The implemented algorithms were tested against a number of TSPLib examples. In the current study, 5 example problems were considered. They are ch130, kroC100, att48, burma14, and bays29. For the ch130 test case, the Max-Min Ant System showed the best result with distance 8045.18. The second place took the Ant Colony System with total distance equal to 10125.01. The worst results showed the Elitist Ant System with distance ot the route equal to 10408.07. Almost the same pattern is observed for the kroC100, and att48. For the burma14, all three algorithms shows almost the same result. For the bays29, the best result showed the Max-Min AS, the second place took the Elitist AS, and the third place belongs to the ACS. So, the best results in the tested problem cases was in 80(%) achieved by Max-Min AS. This is particularly true when the number of nodes is relatively high. The second place is awarded to the ACS. The Elitist AC showed the, in general, third result. The obtained results show that there is no one universal method to solve the TSP problem. In the case when the number of nodes of the route is small, like in burma14, all three methods show almost the same result. When increasing the number of nodes, as in bays29, the worst result is showed by ACS. For the larger number of nodes, the best result is attained by the Max-Min Ant System, while the ACS shows the second result. The third place is given to the Elitist Ant System.

5 Conclusion

There were implemented three methods of ACO,namely, ACS, Elitist AC, and Max-Min AC. Each of the methods was applied to the five TSPLib test cases. The numerical experiments shows that there is no universal method for solving the TSP problem and it depends on the number of nodes in the solved instance. For the case when number of nodes is relatively big (like 100), the Max-Min method gives best results. When the number of nodes is small (like 14), every implemented method shows almost the same result. In the case, when the number of nodes is medium, the Max-Min method is the most effective in terms of distance, while the second place took the Elitist AC.

References

[1] *Applegate DL., Bixby RM, Chvátal V, Cook WJ, The Traveling Salesman Problem, 2006.*

[2] *Eiben AE. Introduction to Evolutionary Computing (Natural Computing Series). Berlin Heidelberg: Springer; 2010.*

[3] *Wang Z., Geng X., Shao Z. An Effective Simulated Annealing Algorithm for Solving the Traveling Salesman Problem. Journal of Computational and Theoretical Nanoscience 2009;6:1680-1686.*

[4] *Basu S. Tabu Search Implementation on Traveling Salesman Problem and Its Variations: A Literature Survey. American Journal of Operations Research 2012;2: 163–173.*

[5] *Dorigo M. 1992. Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italy.*

[6] *Stützle T, Dorigo M. A short convergence proof for a class of ant colony optimization algorithms. IEEE Trans Evol Comput 2002;6(4):358–65.*

[7] *Abdel-Moetty SM. Traveling salesman problem using neural network techniques, The 7th International Conference on Informatics and Systems (INFOS), Cairo, 2010, pp. 1-6.*

[8] *Chowdhury S.,Marufuzzaman M, Tunc H., Bian L., Bullington W. A modified Ant Colony Optimization algorithm to solve a dynamic traveling salesman problem: A case study with drones for wildlife surveillance, Journal of Computational Design and Engineering, 2019, 6(3):368-386*

[9] *Nitesh S. An ACO Approach to Solve a Variant of TSP, International Journal of Advanced Research in Computer Engineering Technology 2012;1(5):222-226.*

[10] *Dorigo M., Di Caro G. Ant colony optimization: a new meta-heuristic. In: Angeline Peter J, Michalewicz Zbyszek, Schoenauer Marc, Yao Xin, Zalzala Ali, editors. Proceedings of the congress on evolutionary computation, vol. 2. IEEE Press; 1999. p. 1470–7.*

[11] *Dorigo M, Birattari M, Stützle T. Ant colony optimization: artificial ants as a computational intelligence technique. IEEE Comput Intell Mag; 2006. p. 28–38.*

[12] *Dorigo M., Di Caro G., Gambardella LM. Ant algorithms for discrete optimization. Artificial Life; 1999. 5(2):137–172*

[13] *Dorigo M, Gambardella LM. Ant colonies for the traveling salesman problem. Bio Syst 1997;43:73–81.*

[14] *Bullnheimer, Hartl RF, Strauss C. A new ranked-based version of the ant system: a computational study. Central Eur J Oper Res 1999;7:25–38.*

[15] *Dorigo M, Maniezzo V, Colorni A. Positive feedback as a search strategy. Technical report, Dipartimento di Elettronica, Politecnico di Milano; 1991.*

[16] *Stützle T, Hoos HH. Max–min ant system. Future Generation Comput Syst 2000;16:889–914.*

[17] *Rechenberg I (1965) Cybernetic solution path of an experimental problem. Royal Aircraft Establishment, Farnborough p. Library Translation 1122*

[18] *Lawler E, Lenstra JK, Rinnooy Kan AHG, Shmoys DB. The travelling Salesman problem. New York: John Wiley Sons; 1985.*

[19] *Hui W. Comparison of several intelligent algorithms for solving TSP problemin industrial engineering. Systems Engineering Procedia 2012; 4:226 – 235*