CS 512 : Final Project
Travelling Salesman Problem Variations

Karan Ashokkumar Pardasani(kp955)
Naishal Chiragbhai Patel(np781)
Himaniben Hareshkumar Patel(hhp46)

December 18, 2021

[1]

karan.pardasani@rutgers.edu
naishal.patel@rutgers.edu
himani.h.patel@rutgers.edu

Contents

1

Introduction

The Travelling Salesman Problem (TSP) is a given combinatorial optimization problem used to find shortest possible route that visits each city exactly once and returns to the original city. The salesman will travel the given cities once in a particular tour and reach the same place where it started. We have to find the order of cities to be visited such as the distance travelled will be minimum. This can be visualized as graph with cities as vertices and path between them as edges. Since it is a NP hard problem, here is no perfect algorithm to solve this problem in polynomial time.

Currently, the only algorithm that gives the exact answer to the Travelling Salesman Problem is the algorithm that goes through all the possible outputs and calculates the path with the minimum weight Hamiltonian Cycle. In this algorithm, each possible solution can be one of the permutations of $1, 2, 3, 4, ..., n$, where n is the number of cities. So, the number of solutions becomes n!. When n gets large, it will be impossible to find the cost of all these solutions in a polynomial time. So, this algorithm of going through all the feasible solutions is not adequate to solve the Travelling Salesman Problem. For this reason, we are required to solve the Travelling Salesman Problem using some meta-heuristic approach. Meta-Heuristic Algorithms are those that do not give the exact optimal solution to the problem, instead they find suboptimal solutions.

We aim to implement different Genetic Algorithms (GA) to find approximate sub-optimal solution to the Travelling Salesman Problem(TSP). In this project, we implement, analyze and compare the Travelling Salesman Problem using three optimization techniques namely:

1. Ant Colony Optimisation(ACO)
2. Black Hole Optimisation(BHO)
3. Particle Swarm Optimisation(PSO).

The time complexity of these algorithms are polynomial. For this project, we used data from this [1] resource. The dataset used in our implementation are DANTZIG42 (set of 42 cities) and att48 (set of 48 US capitals). Also, the dataset contains the minimum tour lengths for each distance matrix which can

be used to analyse the performance of each optimisation technique.

The output of the Travelling Salesman problem is the path the user should travel in a tour. The application will give the path that has the minimum tour according to the optimisation technique (one among the Black Hole, Particle Swarm and Ant Colony Optimisation) chosen by the user.

1.1 Requirement Gathering

In this project, we aim to implement various Genetic Algorithms(GA) to solve Travelling Salesman Problem(TSP) and compare optimal results obtained from each algorithm. Based on the problem constraints, the system can be used to get optimal parameters like: distance, time, scheduling, planning, etc.
Various applications of Travelling Salesman Problem (TSP) are:

1. Computer wiring: On a given computer board there are a given number/subset of pins to be connected with each other. There is a constraint that no two wires must be connected to the same pin. This problem can be visualised as to find Hamiltonian path with a given stating point and ending point. The computer board manufacturer (user) has to test the board to ensure the constraint and make sure the cost is minimum for the desired requirement. The user has to ensure a connection starting from same point and going through all desired set of pins and returning to the same set such as cost is minimum. In this case, pins can be visualised as vertices of graph and the distance between pins can be visualised as edges of that graph.

2. Aircraft gas turbine engines: The requirement of this application is to guarantee uniform gas flow which is guided by nozzle vane at each stage in aircraft. The manufacturer of turbines has to ensure correct placement of vanes to reduce the fuel consumption, vibration, etc. This can be modelled like Travelling Salesman Problem with a special objective function.

3. Transporting Equipment: The farmer/manager (user) of a farm has to transport the soil testing high grade equipment from one place in farm to another.

1.2 Input Format and Output Format

TSP can be represented using weighted graph G = (N, E) where N is the set of n cities and E is the set of edges fully connecting all cities. Each edge $(i, j)E$ is assigned a cost d_{ij}, which is the distance between the cities i and j. Usually, the Euclidean distance is applied to calculate d_{ij} using the following equation:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{1.1}$$

The output format for the algorithms is the Hamiltonian cycle that goes through all the cities exactly once and comes back to the starting point with minimum weight of the path. The output is a list of vertices where the adjacent vertices $(i, i-1)$ represents travelling from vertex v_i and v_{i-1}.

Calculating the Search Space of Travelling Salesman Problem of n cities
Since, there are n cities, the total permutations will be $n!$. And since each configuration of the n cities can be described in $2n$ ways - For example, consider a permutation with n cities. Then the reverse of permutation will represent the same cycle. Also, if we consider the initial permutation and reverse of initial permutation, then there are total more n permutations that will represent the same Hamiltonian Cycle. So, there are total 2n permutation that represents the same Hamiltonian Cycle. Hence, the total Sample Space $|S|$ is:

$$|S| = n!/(2n) = (n-1)!/2 \tag{1.2}$$

1.3 Division of work load and labour

1. Naishal Chiragbhai Patel : Requirement gathering, Formulation of concepts, Data preprocessing, Implementation and Plot generation of Ant Colony Optimization (ACO), Report, Video.

2. Himaniben Hareshkumar Patel : Requirement gathering, Formulation of concepts, Data preprocessing, Implementation and Plot generation of Black Hole Optimization (BHO), Presentation, User Interface.

3. Karan Ashokkumar Pardasani : Requirement gathering, Formulation of concepts, Data preprocessing, Implementation and Plot generation of Particle Swarm Optimization (PSO), Report, Video.

2

Ant Colony Optimisation

2.1 Introduction

Ant Colony Optimization is a Genetic Algorithm(GA), which uses probabilistic techniques to obtain approximate optimal solution of the given graph. All possible solutions are represented by the parameter space which is traversed by artificial ants in order to obtain optimal solution. As real ants leave pheromones along their path to guide other ants towards resources; artificial ants also mimic that behaviour by recording their position (giving probabilistic score) to guide other ants for better solutions.

In order to solve Travelling Salesman Problem(TSP) with given distances $d_{i,j}$ between N cities; we will implement Ant Colony Optimization(ACO) algorithm. Each ant will select next edge along the graph based on probabilistic edge selection. The probability of edge is modified based on the pheromone level. As artificial ants move along the edge; they drop pheromones along the path. Pheromone score is updated by considering the evaporation coefficient and the amount of pheromone deposited by k^{th} ant. Pheromone dropped by each ant is inversely proportional to the cost the respective ant.

Probabilistic Edge Selection: For each ant to select the next edge in the tour; the ant will also consider the pheromone level at each edge along with the length of that respective edge. Each ant will compute probabilities for selecting its next edge. Selection of edges can be modelled as changing of one state to another. In order to move from one state to another ants will consider two factors: **attractiveness** and **past proficiency**. Attractiveness, is the desirability of state transition usually $\frac{1}{d_{xy}}$. Proficiency, is calculated by pheromone dropped while transitioning from one state to another. α and β are the parameter to control the influence of proficiency and attractiveness respectively. The probability is calculate as follows:

$$p_{xy}^{k} = \frac{(\tau_{xy}^{\alpha})(\eta_{xy}^{\beta})}{\sum_{z \in allowed_{xy}} (\tau_{xz}^{\alpha})(\eta_{xz}^{\beta})} \tag{2.1}$$

Pheromone update: Pheromone update considers both evaporation factor and pheromones deposited by the respective ant travelling that edge.

$$\tau_{xy} = (1 - \rho)\tau_{xy} + \sum_{k}^{m} \Delta\tau_{xy}^{k} \tag{2.2}$$

The amount of pheromone deposited while changing states $\Delta\tau_{xy}^{k}$ is given by:

$$\Delta\tau_{xy}^{k} = \frac{Q}{L_k} \tag{2.3}$$

If k^{th} ant moves along that edge xy otherwise 0.

2.2 Summary of Ant Colony Optimization Algorithm

1. For every iteration run the following algorithm

   Loop

2. For each ant do the following

   Loop

3. Select next node to be visited based on probability from equation 3.1.

4. Repeat step 3 until each ant has visited all the cities (nodes).

   End Loop

5. Compute L_k

6. Update pheromone level as shown in equation 2.2 and 2.3.

   End Loop

2.3 Time and Space complexity

2.3.1 Time Complexity

Travelling Salesman Problem can be solved for optimal solution by considering all permutations and hence the time complexity will be n!. But in the implemented Ant Colony Optimization, the time complexity is analysed by referring to [2]. The transition from one state to another depends on several previous state depending on the evaporation factor, the pheromones' effect will last long the given edge. Also, the probability depends on both attractiveness (η) and proficiency (τ) along with the factor of their control of influence (α and β)

If $\alpha = 1$ and $\beta = 1$: T = $\mathcal{O}(n^5 + (1/\rho)n \ln n)$
If $\alpha = 1$ and $\beta = 0$: T = $\mathcal{O}(n^6 + (1/\rho)n \ln n)$

2.3.2 Space Complexity

We have worked on two dataset for ACO: DANTZIG42 (set of 42 cities) , att48 (set of 48 US capitals). The dataset contains 3 columns; first for city index and other two for x and y coordinates. We require $\mathcal{O}(n^2)$ space to store nxn matrices for τ_{xy} and η_{xy}. The list to store edges of the sub optimal solution is in $\mathcal{O}(n)$. Hence the overall space complexity for ACO is $\mathcal{O}(n^2)$.

3

Black Hole Optimisation

3.1 Introduction

The Black Hole Algorithm is a nature-inspired algorithm that mimics the black hole phenomena that occurs in the nature. In nature, black hole occurs when a star of massive size reduces to a small mass and the gravitational power of black hole becomes too high that not even light can escape from it. Anything that crosses the boundary of the black hole will be consumed inside it due to such high gravitational power.

In Black Hole Optimisation, initially a population of possible solutions is generated randomly. These initial solutions are distributed in the problem space. After initialisation, the fitness value of each possible solution is evaluated and the best candidate among the population is chosen to be the black hole and the rest form the normal stars.

After black hole starts absorbing the stars around it, and all the stars starts move towards the black hole. The absorption of the black hole is formulated as follows:

$$x_i(t+1) = x_i(t) + rand * (x_{BH} - x_i(t)) \qquad i = 1, 2, ..., N \tag{3.1}$$

rand is a uniform random number from 0 to 1. $x_i(t)$ and $x_i(t+1)$ are the location of the $i-th$ star at iteration t and $t+1$. x_{BH} is the location of the black hle in the search space. **rand** is the random number in the interval [0,1]. N is the total number of stars.

Since, the stars are moving towards the black hole, it is possible that a star which is different than the black hole achieves a better solution than black hole. In this case, the black hole is replaced by that star and then after the black hole gets updated, then the stars then starts moving towards the new black hole.

During the movement of the stars, there is a chance that the star goes too close to the black hole. The star that gets close to the black hole, gets absorbed in the black hole. So, in our algorithm, if the possible solution gets too close to the black hole solution, then it will get sucked by the black hole. In such a case,

another random solution will be generated and we will start a new search.
We make the black hole absorb the star using the calculation of the event horizon. The rdius of event horizon is calculated as follows:

$$R = \frac{f_{BH}}{\sum_{i=1}^{N} f_i}, \tag{3.2}$$

where f_{BH} is the fitness value of the black hole, and f_i is the fitness value of the $i - th$ star. N is the number of stars.

3.2 Summary of the Black Hole Algorithm

1. Initialise the population of stars with random solutions in the search space.

   Loop

2. For each star, calculate the objective function
3. Select the best star that has the best solution as black hole
4. Change location of the star as per the equation
5. If there is a star with the better solution than the black hole, update the black hole of the system.
6. If the star goes into the event horizon of the black hole, the black hole will absorb the star and hence, new star will be generated in a random location in the search space.
7. The algorithm will terminate when the algorithm finishes a specific number of iterations.

   End Loop

3.3 Space and Time Complexity

Time Complexity

The calculation for time complexity for this algorithm is as follows: **Input**: m: Number of cities, n: Number of stars, iter: Number of Iterations

1. The initialisation will require O(n*m) time
2. For the Loop, for each star we do the following - O(n*iter):
   (a) This will take O(m).
   (b) Selecting the best star will take O(m) time.
   (c) Changing location of star will take O(m) time.
   (d) All the things in the loop will take O(m) time.

Hence, the time complexity will be O(n*m*iter).

Space Complexity

In this Algorithm we store for each star(there are n stars) the solution(takes O(m) space) hence the space required is $O(n * m)$.

4

Particle Swarm Optimisation

4.1 Introduction

To calculate the Travelling Salesman Problem using Particle Swarm Optimisation, we use the concept of Swap Operator and Swap Sequence. The Particle Swarm Optimisation is mimicing the social behaviour that is commonly depicted in organisms. These organisms travel through possible solutions and can store the best solution so far. All the organisms share solutions with each other to calculate the global best solution.

We need to calculate the velocity of each particle to update the solution. So, each particle will update it's velocity in this way:

$$V_{id} = \omega * V_{id} + \eta_1 * rand * (P_{id} - X_{id}) + \eta_2 * rand * (P_{gd} - X_{id}), \tag{4.1}$$

where ω is the inertia factor, X_{id} is current position of the particle, P_{id} is the best solution this particle has reached and P_{gd} is the global best solution of all the particles.

After calculating the V_{id}, we can get the new position in the next iteration as

$$X_{id} = X_{id} + V_{id} \tag{4.2}$$

4.2 Swap Operator

Consider a normal solution sequence of TSP with n nodes:

$$S = (a_i), i = 1...n \tag{4.3}$$

Here we define the Swap Operator $SO(i_1, i_2)$ as exchanging v_{i1} and v_{i2} in solution S. Then we define a new Solution S' as:

$$S' = S + SO(i_1, i_2) \tag{4.4}$$

So, **SO** is a Swap Operator and the **'+'** sign is a special operation used to swap nodes in the sequence.

4.3 Swap Sequence

A Swap Sequence SS is made up of one or more Swap Operators.

$$SS = (SO_1, SO_2, SO_3, ..., SO_n) \tag{4.5}$$

The $SO_1, SO_2, ..., SO_n$ is the Swap Operators that we will apply on a sequence.

Now, Swap Sequence acting on a solution will mean that we apply all the Swap Operators in the order in which they are kept in the Swap Sequence. This can be described by the following formula:

$$S' = S+SS = S+(SO_1, SO_2, SO_3, ..., SO_n) \qquad = (((S+SO_1)+SO_2)+...+SO_n) \tag{4.6}$$

Also, we can see that different swap sequence that act on same solution can produce same result. We can call these set of swap sequences as equivalent set of Swap Sequences. The smallest length of swap sequence that is present in the equivalent class is called Basic Swap Sequence (BSS).

We can also merge two Swap Sequence into one Swap Sequence. We can use the operator $\oplus$ as merging two Swap Sequence into a new Swap Sequence. Suppose there are two swap sequence SS1 and SS2 and if S $\oplus$ S' = S $\oplus$ S_1 $\oplus$ S_2. Then $S' = S_1 \oplus S_2$ is same.

4.4 The Construction of Basic Swap Sequence

If there are two solutions, A and B, and we want to construct Basic Swap Sequence which can act on B to get solution A, we need t define SS = A - B. So, we can solve for A = B + SS. We can calculate SS from the two sequences A and B. For example, consider:
A: (1 2 3 4 5)

B: (2 3 1 5 4)

We need to bring 1 to the first position in B to get the first element of A correct. Therefore the required Swap operator is SO(1,3). After applying SO(1,3) we get
A: (1 2 3 4 5)

B:(1 3 2 5 4)

Now, to get the second element correct, we need to apply the Swap Operator SO(2,3) and as a result we get:

A:(1 2 3 4 5)

B:(1 2 3 5 4)

Hence, we get the first three elements correct, Now to get the fourth element correct, we do the Swap Operator SO(4, 5):

A:(1 2 3 4 5)

B:(1 2 3 4 5)

Hence, we get the following Swap Sequence to get the convert from A to B:

SS -> (SO(1,3), SO(2,3), SO(4,5))

4.5 Update of Velocity for each Particle

$$V_{id} = V_{id} \oplus \alpha * (P_{id} - X_{id}) \oplus \beta * (P_{gd} - X_{id}) \tag{4.7}$$

where $\alpha, \beta[0, 1]$, where α, β are random number from 0 and 1. $\alpha * (P_{id} - X_{id})$ means all Swap Operators are kept in Basic Swap Sequence $P_{id} - X_{id}$ with the probability α. And, all Swap Operators in the BSS $P_{gd} - X_{id}$ are kept in the probability β.

4.6 Algorithm Steps:

1. Initialise each particle with a random solution and a random Swap Sequence called velocity.
2. For all the particles, calculate the next position using the X'_{id} using the following steps:
   (a) Calculate the difference between P_{id} and X_{id} according to the method explained in Section 5.4. Let A is the basic Swap Sequence $A = P_{id} - X_{id}$, where A is Basic Swap Sequence.
   (b) Calculate $B = P_{gd} - X_{id}$, B is also a basic sequence.
   (c) Next, we calculate velocity from equation 5.7, and then we reduce the swap sequence to Basic Swap Sequence.
   (d) Calculate the new solution using $X_{id} = X_{id} + V_{id}$
   (e) Then, we update the P_{id} if the new solution is superior to old P_{id}.
3. Update P_{gd} if there is new best solution, which is superior to old P_{gd}. Then go to Step 2.

4.7 Time and Space Complexity

Time Complexity

The Time complexity of this algorithm is as follows: Input: m: Number of cities, n: Number of Particles, iter: Number of itertions

1. Initialisation:$O(n)$
2. Step -2 :Repeated for $O(n * iter)$ time.

(a) For Step 2(a), $O(m)$ - We iterate through number of cities.

(b) For Step 2(b), $O(m)$

(c) Calculating Velocity from equation 4.7 - $O(m)$

(d) For Step (d) - $O(m)$

(e) For Step (e) = $O(m)$

Therefore, total time complexity is $O(n * m * iter)$.

Space Complexity

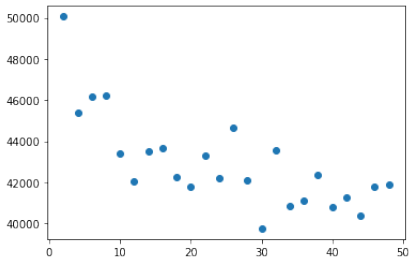
In this Algorithm we store for each particle the solution and swap sequence, hence the space required is $O(n * m)$.

5

Analysis

The actual answer for the plot for which we ran the algorithms is **33,543**.

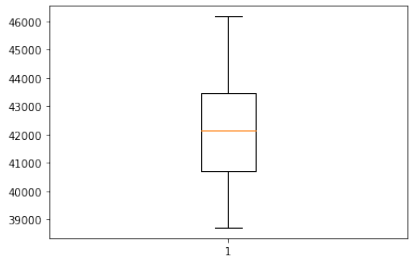
Plot for Ant Colony Optimisation



(a)

Figure 5.1: Plot of Solution of TSP using ACO vs Number of Ants. The x axis represents the number of ants and the y axis represents the cost of the route for Travelling Salesman Problem

The above graph represents the cost of the route for travelling Salesman Problem using the Ant Colony Optimisation. From the scatter plot, we can see that as we increase the number of ants we get more optimal values.

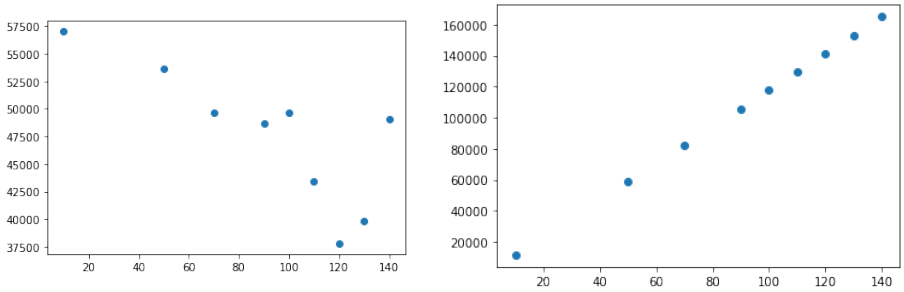


(a)

Figure 5.2: Box Plot of Solution of TSP using ACO.

From the above plot, we can infer that the average of algorithm is around 42500 and also the 25th percentile of the output is at 41000 and the 75th percentile is at 43000. These percentile is pretty close and so this algorithm seems more stable. Also, the minimum answer which is around 39000 is very close to the actual answer **33543**.

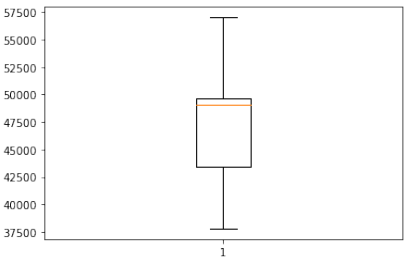
Plot for Black Hole Algorithm



(a) (b)

Figure 5.3: (a) The graph represents the sub-optimal values generated using Black Hole Optimisation. The x axis represents the number of stars and the y axis represents the sub optimal values. (b) This graph represents the amount of search space searched by the algorithm to find the sub optimal value. The x axis represents the number of stars and the y axis represents the number of routes that were processed by the algorithm.

From the above plot, we can see that the global value found by the Black Hole Optimisation improves as we increase the number of stars. Also, the search space for the algorithm is $(n-1)!/2$, hence for $n = 48$, it is around 10^{62}. So, this algorithm searches around 10^5 solutions and it has been able to find a pretty good solution. The minimum answer we were able to find using Black Hole Optimisation is 37500 which is an improvement on the Ant Colony Optimisation algorithm.



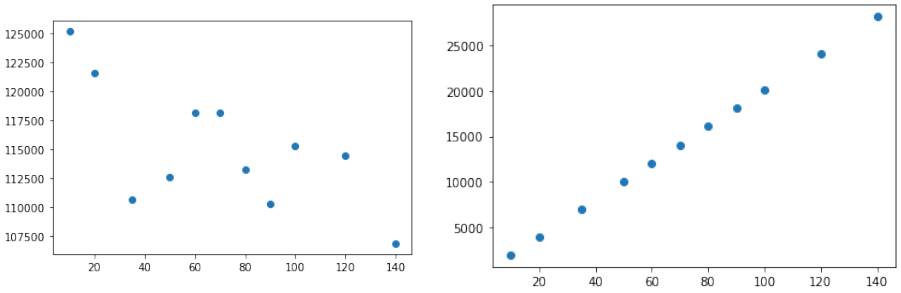
(a)

Figure 5.4: Box Plot of Solution using BHO

From the above plot, we can infer that the average of algorithm is around 49000 and also the 25th percentile of the output is at 44000 and the 75th per-

centile is at 50000. These percentile are very far and so this algorithm seems less stable. Also, the minimum answer which is around 37500 is very close to the actual answer **33543**.

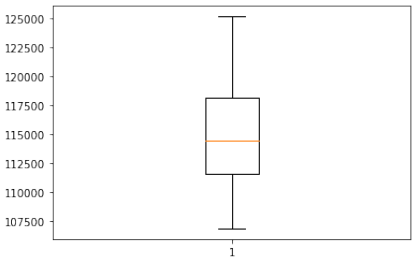
Plot for Particle Swarm Optimisation



(a) (b)

Figure 5.5: (a) The graph represents the sub-optimal values generated using Particle Swarm Optimisation. The x axis represents the number of particles and the y axis represents the sub optimal values. (b) This graph represents the amount of search space searched by the algorithm to find the sub optimal value. The x axis represents the number of particles and the y axis represents the number of routes that were processed by the algorithm.

0



(a)

Figure 5.6: Box Plot of Solution using PSO

From the above plot, we can infer that the average of algorithm is around 112600 and also the 25th percentile of the output is at 112000 and the 75th percentile is at 118000. These percentile are very far and so this algorithm seems less stable. Also, the minimum answer which is around 118000 is very far from the actual answer **33543**.
Hence, from the above analysis we can infer that the Ant Colony Optimisation is more stable than all the algorithms but the most suboptimal answer is given by

Black Hole Algorithm. So, we can see that the worst algorithm is Particle Swarm Optimisation since it gave very huge values for Travelling Salesman Problem.

Bibliography

[1] Dataset - https://people.sc.fsu.edu/ jburkardt/datasets/tsp/tsp.html

[2] Ant Colony Optimisation - Y. Zhou, "Runtime Analysis of an Ant Colony Optimization Algorithm for TSP Instances," in IEEE Transactions on Evolutionary Computation, vol. 13, no. 5, pp. 1083-1092, Oct. 2009, doi: 10.1109/TEVC.2009.2016570.

[3] Black Hole Optimisation - https://link.springer.com/content/pdf/10.1007/s00500-017-2760-y.pdf

[4] Particle Swarm Optimisation - https://ieeexplore.ieee.org/document/1259748