Analysis of the "k-ATSP" problem using hybrid ACO algorithm.

Rakesh Nagaraju
Department of Computer Science
San Jose State University
(Summer 2020)

***Abstract*—In this project, we consider a variant of the well know Travelling Salesman problem (TSP). In particular we consider the k-person Asymmetric TSP known as the 'k-ATSPP' problem as described in [1]. Suppose we have an Asymmetric graph G = (V, A) with two distinguished nodes s, t ∈ V. Our goal is to find the k paths such that it starts from s and ends at t where union of all k paths cover all the nodes only once. Our main result is modified Ant-Colony Optimization (ACO) that includes 3-OPT for optimization and Genetic algorithm (GA) to create new improved paths. We test this algorithm against a TSPLIB dataset against various test cases. Test cases include Single Source/ Single Sink, No Source/ No Sink. We also test this algorithm against the implementation suggested in [1] on test case with No-Source/No-Sink. The results are compared, and time complexity is analyzed. Our results include an algorithm that performs well with large data while maintaining optimal to sub-optimal quality of the results. Finally, we also analyze the reasons for achieving such results.**

***Index Terms*—Travelling Salesman problem, Asymmetric metrics, Ant Colony Optimization, 3-OPT, Genetic Algorithm, Source/ Sink.**

I. Introduction

Travelling Salesman Problem (TSP) is one in every of the foremost studied problems. TSP has many applications associated with vehicle routing problems, logistics, planning, and scheduling. A variation to the present problem is where the Asymmetric graph is taken into account for the multiple salesman and this is referred to as k-person Travelling Salesman Problem (k-ATSPP) [1]. Given a Asymmetric graph G = (V,A) with two distinguished nodes s, t ∈ V and a positive integer k. The idea is to k paths from s-t such that union of all k paths covers all nodes in {V} and also the total cost is minimum. The authors in [1] structured this problem and also proposed a bicriteria approximation solution to solve it. In this project, we study efficient and recent approaches for solving Asymmetric TSP and m-TSP problems. We then propose a hybrid approach to the k-ATSPP problem that is combination of Ant colony optimization, 3-OPT exchange and Genetic Algorithm. We then test this approach on TSPLIB asymmetric data and finally compare the results against a dynamic programming approach. Experiments with various test cases are also considered such as: No Source, Single Source/Single Sink. The rest of this section is as follows: Section II provides the Literature review, details of the current solution and also mentions the strengths and weakness. Section III explains the proposed methodology, Section IV provides implementation details and results. Section V concludes the paper and also provides future work.

II. Literature Review

A variation to the ATSP problem is proposed in [1] where the authors state the problem as: given an asymmetric metric G = (V, A) with two distinguished nodes s, t ∈ V and a positive integer k. The main aim is to find k paths of minimum total cost from s to t, whose union spans all nodes. This is called as the k-Person Asymmetric Traveling Salesman Path problem (k-ATSPP). The authors in [1] also propose Linear programming (LP) relaxation-based bicriteria approximation algorithm for solving this problem.

There are many proposed algorithms for solving the regular Travelling Salesman as well as its variants like: multiple TSP, Symmetric and Asymmetric TSP. However, k-ATTSP has very few proposed solutions, and [3] is one such paper where Essani et al., designs an algorithm for using Colored Petri Nets (CPN) - a formal mathematical modeling language, for modeling the asymmetric Multiple Traveling Salesman Problem. Firstly, a mechanism is proposed for mapping city, depo, salesman, and time from Multiple Traveling Salesman Problem to place and transition in a Colored Petri Net, and then the proposed algorithm transforms any given asymmetric Multiple Traveling Salesman Problem into a Colored Petri Net model. The transformed model in CPN is simulated in CPNTools to calculate different optimization objectives and is formally verified through reachability analysis to ensure that various properties hold true for the complete state space of the model. The proposed solution always guarantees a feasible solution and is flexible enough to solve the multiple TSP problem on both asymmetric or symmetric cost matrices. The simulation results for this approach can be seen in Table II below.

[4] considers the multiple TSP problem and proposes a new hybrid approach called AC2OptGA, which is a combination of three algorithms: Modified Ant Colony, 2-Opt, and Genetic Algorithm. Ant Colony-based algorithm is used to generate solutions on which the 2-Opt edge exchange algorithm is

applied to enhance the obtained solutions. A Genetic Algorithm is then used to again improve the quality of the solutions. ACO is used to generate a population of good feasible solutions for the regular TSP. Then, 2-Opt is applied to improve these solutions. After that, every TSP solution is randomly broken down into a multiple TSP solution. Then, one step of GA is applied to the previously obtained solutions. In this step, suitable selection, crossover, and mutation operators are applied to improve the multiple TSP solutions. The Pheromone map is then updated to guide the next iteration. The proposed approach provides better solutions for multiple TSPs with large and medium-sized data instances and competitive results for small-sized data instances on the TSPLIB when compared with the other approaches for solving the multiple TSP. The results of these experiments can be seen in TABLE III below.

An algorithm to solve the Asymmetric TSP is proposed in [5] where Rahman et al., proposes a dynamical route construction algorithm to solve both versions of a TSP i.e., symmetric and asymmetric TSP's. This algorithm first produces possible probes based on the probe concept, then it keeps the potential probes with the help of the proportion value. Finally, it utilizes local search operators to improve potential probes. This quality improvement continues until all of the steps are completed. Finally, the best complete route seeks out from the good probes found in the last step. It is demonstrated by the experimental results that the proposed algorithm can reach the optimal point for a mentionable number of instances and usually provide an approximate solution closer to the best-known solution within a reasonable CPU running time. In addition, the algorithm generates the results that are superior to the best-known results reported by the data library in some cases and outperforms state-of-the-art algorithms for the selected instances which can be seen in Table IV below.

A. *Current Solution:*

To solve the k-ATSPP problem, a bicriteria approximation algorithm is proposed in [1] that considers an asymmetric metric G = (V, A), whose total cost is within a bounded ratio of the optimum value of LP relaxation. The LP relaxation for k-ATSPP is similar to the LP relaxation for ATSPP and is given as follows:

$$\text{minimize} : \sum_{e \in A} d_{uv} \cdot x_{uv}$$

$$\begin{aligned} \text{subject to} : & \; x(\delta^+(v)) = x(\delta^-(v) = 1 && \forall v \in V - \{s, t\} \\ & x(\delta^+(s)) = x(\delta^-(t)) = k \\ & x(\delta^-(s)) = x(\delta^+(t)) = 0 \\ & x(\delta^+(S)) \geq 1 && \forall \{s\} \subseteq S\,(V) \\ & x_{uv} \geq 0 && \forall\, u,v \in A \end{aligned}$$

This algorithm is also parameterized by a positive integer b where different bicriteria approximations guarantee result from different choices of b. The algorithm returns approximately k paths from s to t as the output. The authors consider, that in a k-path/cycle cover F, the flow F_{uv} across any arc $uv \in A - \{st\}$ is either 0 or 1 and $F_{st} \leq k$. F is a multiset of arcs and is decomposed into k paths from s to t and a collection of cycles where every $v \in V - \{s, t\}$ lies on exactly one path or exactly one cycle. A minimum-cost k-path/cycle cover is found using a standard reduction to minimum weight perfect matching in a bipartite graph with $n + k - 2$ nodes on each side which is similar to cycle covers from [9] and path/cycle covers from [10] and [11]. The algorithm is divided into two parts.

The first part is a "for" loop that runs for $\Theta(b \log n)$ iterations. In every iteration we find minimum-cost k-path/cycle cover F on W, the set of vertices not discarded. Then, the authors remove circulations or flows to make the graph acyclic, also discards nodes from the cycles by taking the union of the current k-path/cycle cover and paths from previous iterations. At the end of the first part we have $\Theta(bk \log n)$ paths of cost from s to t whose union is acyclic and covers all remaining nodes.

The second part of the algorithm will assemble only $(k + k/b)$ paths that cover all remaining nodes using arcs from the $\Theta(bk \log n)$ paths. First, we find $k' \in [k, k + k\,b\,]$ $s - t$ paths whose union covers all nodes in W call it P and B be a simple $t - s$ flow. Then, we select $C' \leftarrow P + C + B$ (C is the discarded nodes from each iteration). Next, we deduce a Eulerian circuit K using each arc uv exactly C'_{uv} times. Finally, a path decomposition of K yields the required paths: P1,.....Pk. The results of the experiments performed in [1] can be seen in TABLE I below.

TABLE. I. Results

K-value	K = 1	K = 2	K > 2
Complexity	O(log n)	O(log n)	O(k log n)
No of Paths	2k	2k	k

TABLE. II. Results

Optimization Objective	Case A (Time Units)	Case B (Time Units)	Case C (Time Units)
miniSUM	18	18	NA
miniMAX	8	7	15

TABLE. III. Results

				AC2OptGA
Instance	n	u	BS	AS
Pr76	76	20	159289	163120
Pr152	152	40	127520	132057
Pr226	226	50	161084	163235
Pr299	299	70	77810	80247
Pr439	439	100	149675	153199
Pr1002	1002	220	351371	355790

TABLE. IV. Results

Sl No.	Instances	No. of cities	Optimum	Our results	Error (in %)	CPU time (in seconds)
Asymmetric Traveling Salesman Problem (ATSP)						
1	br17	17	39	**39**	0.000	0.0089 ± 0.0027
2	ftv33	34	1286	**1286**	0.000	0.1910 ± 0.0068
3	ftv35	36	1473	**1473**	0.000	0.3845 ± 0.0099
4	ftv38	39	1530	**1530**	0.000	10.9092 ± 0.2784
5	p43	43	5620	**5620**	0.000	0.1820 ± 0.0069

B. *Strengths:*

The proposed solution in [3] always guarantees a feasible solution and is flexible enough to solve the multiple TSP problem on both asymmetric or symmetric cost matrix.

In [4], the probability of a fall in local minima is minimized by exploiting the strengths of global search-based algorithms (ACO, GA) as well as a local search (2-Opt).

In the experiments conducted in [4], AC2OptGA outperforms other methods when the number of salesmen is less than 10. The quality of the obtained solution is improved as ACO and GA use random-based search algorithms for exploring the state space. In [4], a better solution is always guaranteed as the three methods together iterate until a better solution is produced.

In [5], the proposed algorithm always produces the same result in every run as there is no randomness in the procedure. Also, no fine-tuning of parameters are required except for the filtering proportion value. Furthermore, the proposed algorithm in [5] can solve the asymmetric problem with almost equal flexibility of symmetric problems except in the first step. The best approximation for k-ATSPP using exactly k paths has an approximation guarantee of O(k log n).

The best approximation for general k-ATSPP, when k = 1 or when k = 2 is proved to be O(log n). Since this is the only proposed approach for the k-ATSPP problem, the obtained results are better than a brute force search at the least.

C. *Weakness:*

In [3], the utility of the mapping schemes comes into play as constraints are added to multiple TSP, and thus finding a feasible solution becomes a complicated task. In [4] when the number of salesmen is greater than 10, AC2OptGA produces an average performance. With certain new approaches producing a wide range of high-performance solutions with small differences between the best, average, and worst. We can thus say that there is still room for improvement by incorporating smart techniques to build better solutions for multiple TSP.

Also, the algorithm proposed in [4], only guarantees a solution for Asymmetric version of the TSP with one salesman. Further experimentation is required to determine if this can be applied to our variant of the problem (k-ATSPP).

We saw that the proposed algorithm in [5] can solve the asymmetric problem with almost equal flexibility of symmetric problems. However, can this algorithm be used to solve the asymmetric version of the multiple TSP, is a topic of further analysis. For different values of k, we obtain different complexities. On one extreme, we get O(log |V|)-approximation that uses up to 2k paths and on the other end, we get O(k log |V|)-approximation that uses exactly k paths. However, further analysis is required to determine if we can reduce the dependency of k on the complexity.

One weakness in the algorithm that the authors in [1] also agree is, that rather than minimizing the total cost of all paths, an ideal way would be to minimize the cost of the most expensive path. This can be thought of as minimizing the total time it takes for agents moving simultaneously to visit all locations. Regular Travelling Salesman Problem can be solved in Poly-logarithmic time and the proposed solution does not give Poly-logarithmic time. Further experimentation is required to see if we can design an algorithm that solves the k-ATSPP problem in Poly-logarithmic time.

III. METHODOLOGY

There are many proposed solutions for solving the regular TSP problem. However, very few solutions are proposed to solve k-ATSPP. In this project, I would like to use the ACO algorithm and its features to find the shortest path of an Asymmetric graph by including a local search algorithm and a Genetic Algorithm (GA). Ant colony optimization algorithm (ACO) is one such probabilistic technique that can be used for solving computational problems and finding the shortest paths through graphs by further reducing them. The fundamental idea of ant heuristics is based on the behavior of real ants that always finds the shortest paths from their nest to food by communicating via pheromone trails.

Artificial Ants are inspired by the behavior of real ants. The pheromone-based communication can be seen here, where the ants follow the path based on the pheromone trail left by the previous ants. These ants locate optimal solutions by moving through all possible solutions. These 'ants' record positions and quality of their solutions, so that in later iterations, the next set of ants can locate better solutions. Combinations of Artificial Ants along with local search algorithms such as: 2-OPT or 3-OPT have been used to solve many optimization tasks involving graphs, e.g., vehicle routing and internet routing.

Local search algorithms are used to optimize the solutions obtained and GA is applied to divide the optimal solution among k- traveling salesmen such that their union spans all cities. During every iteration, pheromones are calculated and updated. Finally, the paths yielded are returned as the final output.

The ACO algorithm is a greedy based approach where the quality of the solution is considered rather than the time for execution. Hence, the main aim of this project is to obtain better solutions rather than improving time complexity. The methodology can be seen in Figure 1 below.

Fig.1. Methodology

A. *System Architecture Diagram*

The system architecture diagram for the implementation plan can be seen in Figure 2 below, which describes the overall methodology of this project and also contains details of the new and old components used.

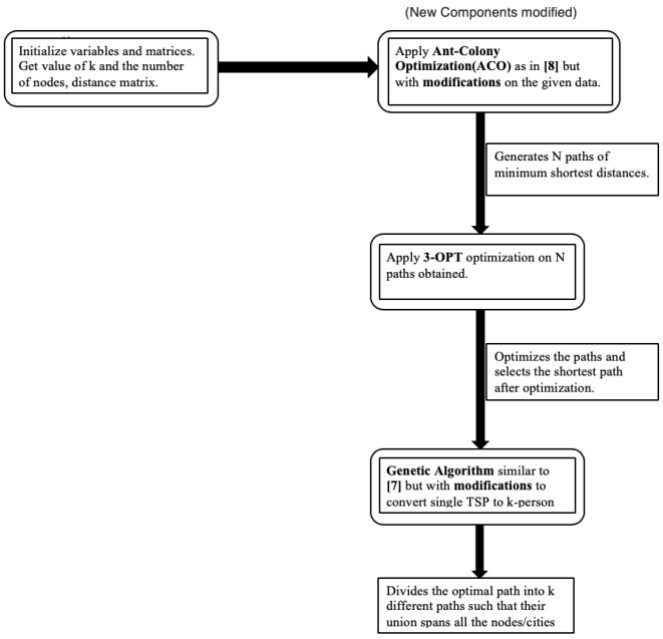

Fig.2. System architecture diagram

B. *Data Source*

To run our code we need an Asymmetric distance matrix with 'n' number of cities and 'm' number of salesmen. Therefore, we obtain the "Asymmetric traveling salesman problem (ATSP)" dataset with varying number of cities from the library TSPLIB that contains literature-standard benchmarks for solving TSP and m-TSP. This dataset contains 19 text files with the number of cities varying from 17 to 443. The number of salesmen 'm' is provided as a parameter while running the code. In this project, we use random range among the 19 text files for testing the code and benchmarking the results. The file can be found on the webpage: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/atsp/. The contents of the file can be seen in the Table V below.

TABLE. V. Contents of TSPLIB

NAME (name of the file)	EDGE_WEIGHT_TYPE (weight type of the edge such as: explicit)
TYPE (the type of the file such as ATSP)	EDGE_WEIGHT_FORMAT (weight format of the edge such as Full Matrix)
COMMENT (such as 17 city problem)	EDGE_WEIGHT_SECTION (the asymmetric distance matrix).
DIMENSION (dimension of the distance matrix such as 17)	

C. *Testing and Benchmarking plan*

We test both the current solution and my proposed solution on the TSPLIB library data with the Asymmetric matrix mentioned in the section above.

To test our results, I tried to implement the current solution for the k-ATSPP proposed in [1] using python. However, this was not possible and we implement Dynamic Programming approach based on Held-Karp algorithm for comparing the results.

Next, we run both these implementations against the data source by varying the value of salesmen (m). The different number of salesmen are validated as different test cases and the results will be stored. The results include the shortest path, distance, and execution time.

Finally, we compare the results between the approaches and my implementation. The main aim of this project is to improve the quality of the results. Thus we compare the quality of the results such as the path with the least distance spanning all cities along with checking the time complexities.

IV. Implementation Details and Results

In this section first, we see details of how the algorithm works, then see the implementation flow and finally analyze the results.

A. *ACO*

We generate an 'N' set number of Ants placed on starting nodes and they travel to and from that node, visiting all nodes, using various available paths. These ants also deposit pheromone on return. Thus pheromone deposition is more on shorter distances than the longer ones. An individual ant decides on the next node to visit based on the level of pheromone on the path and the distance to the nearest node. We maintain a matrix for the pheromone.
Next node to visit is calculated by:

pheromone ** α * (distance) ** β,

where alpha and beta are weights for pheromone and distance respectively. In our implementation, we spread pheromones only on the shortest path found during each iteration and evaporate pheromones from all other edges. Figure 3. Gives an visual idea of ACO.

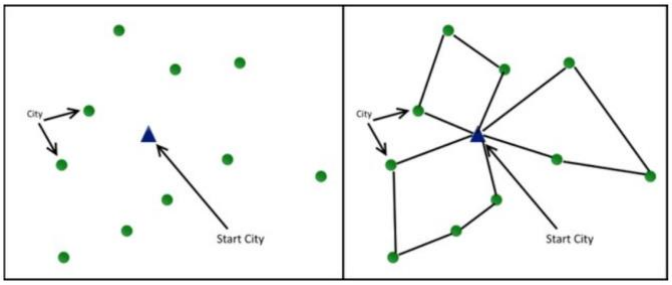

Fig. 3. Ant-colony

B. *3-OPT*

ACO has the tendency to get stuck in a local optimum, hence a local search algorithm is required. Hence, use 3-OPT optimization. So, once, we get a complete path from ACO, our next step is to optimize this path.

To achieve this, we can use 2-OPT as in [4]. However, the problem with 2-OPT is that it always reverts the edges selected. But, this causes a problem in the case of asymmetric matrices. Hence, we use 3-OPT as this does not revert the path and is thus appropriate for asymmetric TSP.
The 3-OPT analysis involves interchanging 3 edges in a graph to create 3 sub-tours. Then all the various paths for reconnecting are analyzed and then select the most optimum one. This process is repeated until all possible combinations have been tried out in the graph. On the successful execution

of this step, we now have a minimum path that is optimized by 3-OPT. Figure 4. Gives an visual idea of 3-OPT.

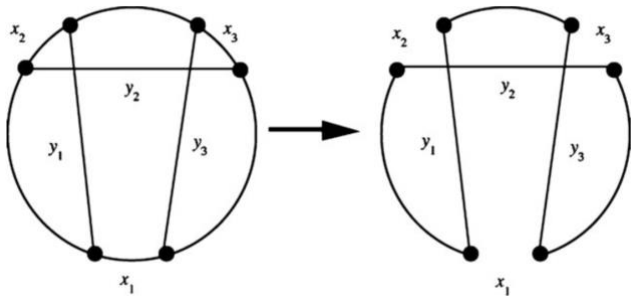

Fig.4. 3-OPT

C. Genetic Algorithm

Our next step is to divide the path among k-traveling salesmen. To do this, we use Genetic Algorithms similar to [4]. Here we transform every TSP solution into a k- person TSP. This is done randomly as we select (k – 1) points to cut in the TSP solution and then assigning each part of the TSP solution to a salesman.

Multi-chromosome representation technique suggested by Kiraly et al. in [7], is used here as it is proved that this representation produces the smallest complete search space by eliminating redundancies.

Each chromosome represents a salesman tour with a set of unique cities to be visited. The size of the search space is n!. The depot is fixed for all salesmen and the Genetic Algorithm is applied to each group of k solutions in P.

The fitness function calculates the cost of each k solution as the summation of distances between their nodes. Then the least expensive solution is selected and we apply different crossover and mutation operators proposed in [7] as seen in Figure 5, 6, 7 respectively.

Finally, we return the optimal k paths for the optimal solution generated by ACO and optimized by 3-OPT. The Figure 5, 6, 7 below describes the GA operations in high level.

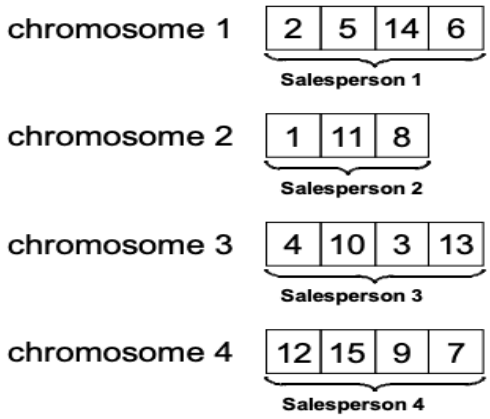

Fig. 5. Genetic Algorithm multi-chromosome

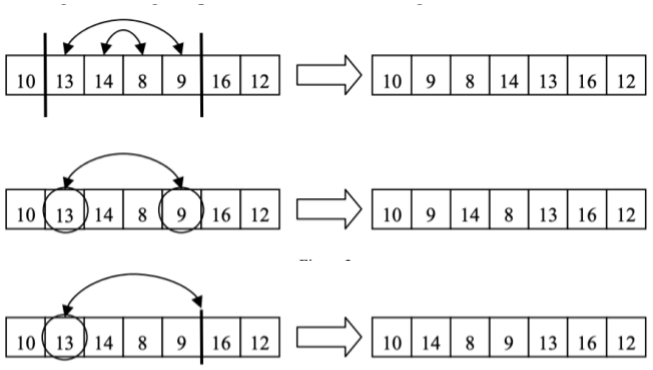

Fig. 6. Genetic Algorithm Operation

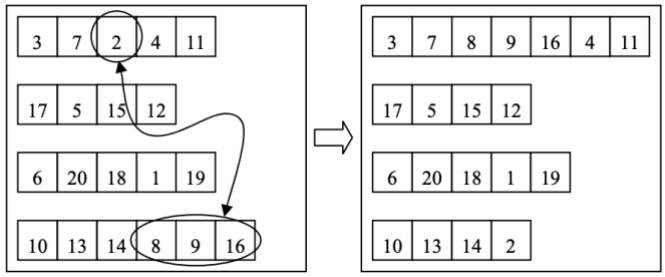

Fig. 7. Genetic Algorithm Crossover

D. The Algorithm

From previous sections we saw how we proposed an algorithm using ACO, 3-OPT and GA. Now we see in details how they work. Figure 8. below gives the complete algorithm and the Figure 9. gives a clear flow of the algorithm.

```
procedure main():
   Initialize the variables and set parameters
   For N Iterations:
      Generate Population P of single paths from s-t using ACO

      Apply 3-OPT to optimize the obtained path

      Split the path randomly among k- person to get k-paths from s-the

      Sa = min cost path in the Population

      Find the best solution and apply 7 Genetic Algorithm
       Operations to generate 8 new paths

      Add these path to Population

      Calculate Fitness Again

      Sb = new min cost path in the Population

      Evaporate pheromones on edges A

      if Sa < Sb
        Deposit pheromones on Sa edges
        current_solution = Sa
      else
        Deposit pheromones on Sb edges
        current_solution = Sb
      end-if
      if best_solution > current_solution:
         best_solution = current_solution
      end-if
      deposit pheromone on best_solution edges
   end-while
End
```

Fig. 8. The Algorithm

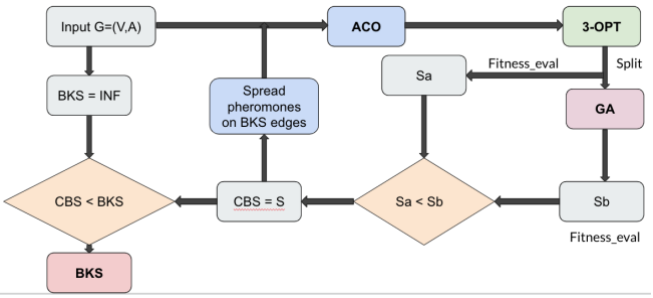

Fig. 9. Complete flow of the Algorithm

E. *Test cases and Results*

Our code is run on MacBook Pro, 2.7 GHz Intel Core i5 Processor, 8 GB RAM, 1536 MB Intel Graphics.

Within 19 available TSPLIB text files with the number of cities varying from 17 to 443, we used low, high and average number of test case files for testing the code and benchmarking the results. Also for low case scenarios we synthesized a distance matrix of size 7. Parameters to run the code include:

K = Number of Salesman
(s,t) = optional, starting and endpoint
D = Distance Matrix
Iteration = No of Iterations
Ants = Number of Ants
Decay = decay rate (0.95)
Alpha , Beta = weights (1)

Our implementation was tested with following test cases:

1.) K- Value: k = 1, k = 2 and k > 2.
2.) Case 1: Random tours for each salesman
3.) Case 2: Round tours for every salesman (Starting and end-point are same for all k)
4.) Different value of n(cities) = 7, 17, 36, 48,170.

The results for the above testcases can be seen in Table VI, VII below:

TABLE. VI. Results

		n = 7		n =17		n = 38	
		Total Distance & Time in seconds					
k = 1	No source	25	0.06	56	0.2	2979	2.99
	Single source (4,1)	44	0.25	44	0.25	3062	1.4
k = 2	No source	32	0.07	65	1.5	5116	2.9
	Single source (0,3)	45	0.08	167	2.8	6466	0.7
k = 4	No source	14	2.08	203	4.4	10093	9.6
	Single source (0,4)	16	2.26	199	0.96	**6593**	**26.8**
k = 5	No source	116	0.07	197	1.3	11981	5.48
	Single source (2,3)	171	0.06	197	5.1	10121	1.16

TABLE. VII. Results

		n = 48		n = 170	
		Total Distance & Time in seconds			
k = 1	No source	3753	5.2	14399	88.6
	Single source (4,1)	3662	1.17	12254	30.0
k = 2	No source	6688	1.14	20162	30.4
	Single source (0,3)	6367	5.6	19660	64.14
k = 4	No source	9332	24.8	16778	**89.5**
	Single source (0,4)	10310	1.9	22528	42.9
k = 5	No source	12185	2.5	40451	83.7
	Single source (2,3)	10072	4.34	22233	40.40

We also made modifications to an existing Dynamic approach (called the Held-Karp algorithm), to handle the k-ATSPP problem. This Dynamic algorithm handles the No-Source variant of the problem, i.e,. the system randomly generates the (start, end)-points for calculating the tour. These random points will be displayed to the user as the Output along with the shortest path and the total cost. We compare the results of both the approaches on a common grounds by executing both approaches on the same datasets and also compare the quality of the results along with the run-time of execution. The results for this comparison can be seen in the TABLE VIII below:

TABLE. VIII. Comparison Results

		n = 7		n =17		n = 38	
	Random source (s,t)	Total Distance & Time in seconds					
k = 1	Modified ACO (0,2)	**10026**	**0.6**	**10044**	**9.1**	2711	16.05
	DP (0,2)	**10025**	**0.001**	**10033**	**4.8**	2657	> 5min
k = 2	Modified ACO (0,2)	**30**	**0.113**	131	31.6	5116	2.9
	DP (0,2)	39	0.001	92	4.8	5002	> 5min
k = 4	Modified ACO (0,2)	260	0.19	420	1.07	10093	9.6
	DP (0,2)	101	0.001	160	4.7	10193	> 5min
k = 5	Modified ACO (0,2)	191	0.15	608	0.99	11981	5.48
	DP (0,2)	101	0.001	159	4.6	11891	> 5min

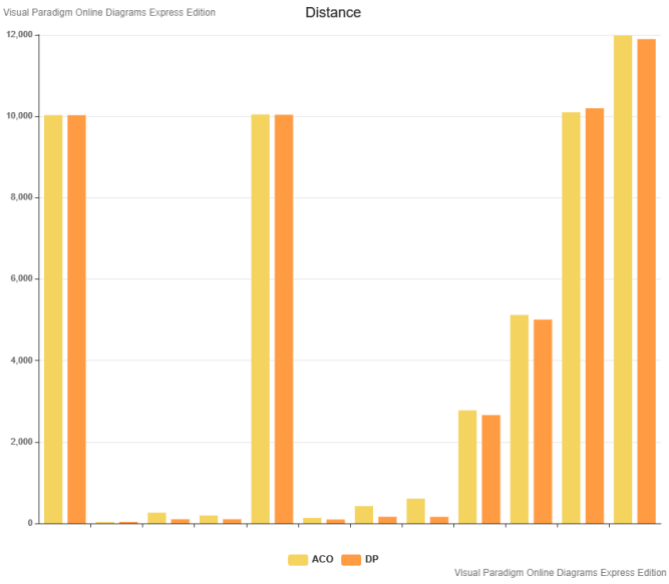

Fig. 10. Comparison graph

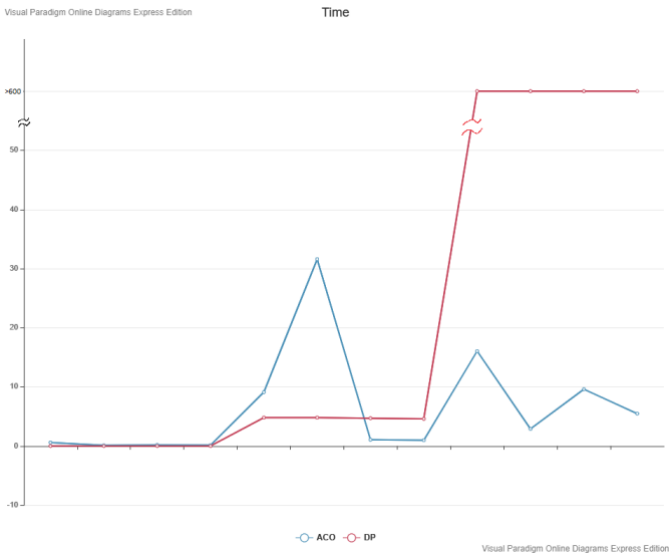

Fig. 11. Comparison graph

Some screenshots of the project is situated at the end of this section in Fig 12, 13, 14 to have a look. To find all the screenshots kindly refer the Git-hub link of this paper.

As we can from the above Table VII, VII and VIII and graphs in Figure 10, 11, we get good time complexities with our ACO implementation, whereas Dynamic programming time complexity increases exponentially for cities > 30.

So, in terms of time performance, ACO is better than the Dynamic Programming. On comparison, the quality of results shows that ACO performs better in some cases and in other cases DP performs better. This might be due to the randomness present in the ACO implementation. In particular, because of randomization while splitting the optimal path among salesman, in 3-OPT and as well in GA mutation and Crossover operations.

One way to optimize this code would be to:

1.) Instead of running for N no of Iterations, it would be ideal to have a termination condition.
2.) Reduce Randomization in code.
3.) In 3-OPT only select the edges that has highest edge distance in the entire path.

```
Enter '0' to continue, '3' to Exit : 0
Enter filename : fasym7.txt
Enter k (Salesman): 2
Want to specify end points? Y/N : N
Selected (s,t) is :  (4, 2)
Enter '1' to run with default parameters or '2' for setting the parameters : 1
Best_Solution : ([[4, 1, 0], [6, 5, 2]], 38)
Elapsed time :  0.13720965385437012
For values given File : fasym7.txt  k =  2  start-end points =  (4, 2)
```

Fig. 12. Screenshot-1 (k =1)

```
Enter '0' to continue, '3' to Exit : 0
Enter filename : br17.txt
Enter k (Salesman): 4
Want to specify end points? Y/N : N
Selected (s,t) is :  (11, 9)
Enter '1' to run with default parameters or '2' for setting the parameters : 1

Best_Solution : ([[11, 5, 14, 8], [3, 1, 6, 15], [11, 0], [13, 15, 10, 9]], 144)
Elapsed time :  2.0878939628601074
For values given File : br17.txt  k =  4  start-end points =  (11, 9)
```

Fig. 13. Screenshot-1 (k =2)

```
Enter '0' to continue, '3' to Exit : 0
Enter filename : fasym7.txt
Enter k (Salesman): 1
Want to specify end points? Y/N : N
Selected (s,t) is :  (3, 1)
Enter '1' to run with default parameters or '2' for setting the parameters : 1
Best_Solution : ([[3, 6, 4, 0, 5, 1]], 44)
Elapsed time :  0.19304704666137695
For values given File : fasym7.txt  k =  1  start-end points =  (3, 1)
```

Fig. 14. Screenshot-1 (k = 4)

V. Conclusion and Future Work

In this paper, we encountered a new variant of the well-known Travelling Salesman Problem, the “k-ATSPP”. We saw a Bi-criteria approximation to solve this problem based on the LP relaxation of Held-Karp algorithm. We saw various literatures that tries to deal with a similar problem.

Next, we encountered 3 new algorithms: ACO, 3-OPT and GA. We also proposed a solution using these algorithms to solve the k-ATSPP. We successfully implemented the proposed algorithm.

Furthermore, we experimented with various test cases such as: No Source, Single Source/Single Sink and achieved good results with good time complexity. Additionally, we compared our approach results with modified dynamic programming approach.

We found that our algorithm has better time complexities than the dynamic programming especially on large datasets. Dynamic programming takes >5 mins for data with more than 30 cities. However, the quality of the results vary within the proposed approach. We also analyzed why this the case and a possible solution. Finally, not many solutions are available for k-ATSPP, so a different approach for solving k-ATPP is obtained by this project.

As a part of future work, I would like to:

1.) Modify this approach to handle multiple source/ Multiple Sink scenario.
2.) Implement the algorithm in [1] for better comparison among the results
3.) Would try an approach to minimize the cost of the most expensive path as seen before.
4.) Implement Artificial Bee-colony optimization instead of ACO with 3-OPT and GA.

REFERENCES

[1] Friggstad Z. Multiple Traveling Salesmen in Asymmetric Metrics. In: Raghavendra P., Raskhodnikova S., Jansen K., Rolim J.D.P. (eds) Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. APPROX 2013, RANDOM 2013. Lecture Notes in Computer Science, vol 8096. Springer, Berlin, Heidelberg (Revised 2018).

[2] Wei Gao. New Ant Colony Optimization Algorithm for the Traveling Salesman Problem "International Journal of Computational Intelligence Systems". (Published: 2020/01)

[3] Essani, F.H.; Haider, S. An Algorithm for Mapping the Asymmetric Multiple Traveling Salesman Problem onto Colored Petri Nets. Algorithms 2018, 11, 143.

[4] Youssef Harrath, Abdul Fattah Salman, Abdulla Alqaddoumi, Hesham Hasan & Ahmed Radhi (2019) A novel hybrid approach for solving the multiple traveling salesman problem, "Arab Journal of Basic and Applied Sciences", 26:1, 103-112, DOI: 10.1080/25765299.2019.1565193.

[5] Rahman M.A., Ma J. (2019) Solving Symmetric and Asymmetric Traveling Salesman Problems Through Probe Machine with Local Search. In: Huang DS., Bevilacqua V., Premaratne P. (eds) "Intelligent Computing Theories and Application". ICIC 2019. Lecture Notes in Computer Science, vol 11643. Springer, Cham.

[6] Viswanath Nagarajan, R. Ravi. Poly-logarithmic Approximation Algorithms for Directed Vehicle Routing Problems. Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 2007, Volume 4627.

[7] Kiraly, A., & Abonyi, J. (2011). Optimization of multiple traveling salesmen problems by a novel representation based genetic algorithm. In: Intelligent computational optimization in engineering (pp. 241–269). Berlin, Germany: Springer.

[8] AntColonyOptimization (2018) by Kirill Temlyakov. In: github.com/Akavall/AntColonyOptimization

[9] A. Frieze, G. Galbiati, and F. Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. Networks, 12:23–39, 1982.

[10] Z. Friggstad, M.R. Salavatipour, and Z. Svitkina, Asymmetric traveling salesman path and directed latency problems. Proc. SODA, 2010. Additional results appear in arXiv, manuscript number 0907.0726v1.

[11] F. Lam and A. Newman. Traveling salesman path problems. Math. Program., 113(1):39–59, 2008.