Comparison of Approximate Approaches to Solving the Travelling Salesman Problem and its Application to UAV...

Article · January 2015

DOI: 10.14323/ijuseng.2015.1

3 authors:

Anoop Sathyan
University of Cincinnati

Nathan Boone
University of Cincinnati

Kelly Cohen
University of Cincinnati

Research Article

IJUSEng – 2015, Vol. 3, No. 1, 1-16
http://dx.doi.org/10.14323/ijuseng.2015.1

Comparison of Approximate Approaches to Solving the Travelling Salesman Problem and its Application to UAV Swarming

Anoop Sathyan, Nathan Boone and Kelly Cohen ✉
University of Cincinnati, USA.

Abstract: Sathyan A, Boone N and Cohen K. (2015). Comparison of approximate approaches to solving the Travelling Salesman Problem and its application to UAV swarming. *International Journal of Unmanned Systems Engineering*. ***3***(1): 1-16. The Travelling Salesman Problem (TSP) is a widely researched Non-deterministic Polynomial-time hard optimization problem with a range of important applications in a wide spectrum of disciplines including aerospace engineering. In this paper, a comparison of different approaches to solve the TSP and also its application towards swarming of UAVs is considered. The objective of the TSP is to determine the optimal route associated with the shortest tour connecting all targets just once. Genetic Algorithms (GA) are one of the most widely applied techniques for solving this class of optimization problems. Two other techniques, 2-opt and Particle Swarm Optimization, are used and the results are compared with those obtained using GA. The comparison is made for different numbers of targets, using salient figures of merit such as computational time required and the cost function which is the minimum solution (distance) obtained. Results show that the 2-opt approach with the closest neighbour as initial starting point for the search yields superior performance. In the Multiple Travelling Salesman Problem, we propose a cluster-first approach which allocates each specific UAV to a subset of targets. The 200 targets are divided into four clusters corresponding to the four UAVs and then TSP algorithms like 2-opt and GA are employed to solve each cluster. This approach drastically reduces the computational time and also gives much better results than the conventional technique of directly applying GA over the 200 targets.

Keywords:
Clustering
Genetic Algorithms
K-means
Multiple Travelling Salesman Problem
Particle Swarm Optimization
2-opt

I. Introduction

The TSP finds a wide variety of applications. Some of the main applications are in the field of UAV swarm routing [1,2], PCB drilling [3], and the warehouse order picking problem [4]. As a result of increasing research in the field of robotics and autonomous control systems, the UAV industry has seen a rapid rise in the past two decades. The flight of a UAV is controlled by either its onboard computers or by commands transmitted from the ground station. The computational power that is available today can be harnessed to control thousands of UAVs from

Correspondence
School of Aerospace Systems
College of Engineering & Applied Science
Rhodes Hall - Room 735
PO Box 210070
Cincinnati, OH 45221-0070
Kelly.Cohen@uc.edu

a ground station. In order to get a perspective on the scale of this computational mileage, UAVs attached with cameras could be used to monitor an entire city from a single ground station. Other than aerial surveillance, UAV swarms are also used for air combat, communications/data transfer, transporting goods, and more. Most of the applications of UAVs require knowledge of the geographical locations of the different targets or cities so that the UAVs can autonomously cover the targets. This can be done optimally if the UAVs can be programmed to choose the shortest path that covers all the targets, which is known as path planning.

One of the most widely researched path planning problems is the Travelling Salesman Problem (TSP) or its multi-agent version, the Multiple Travelling Salesman Problem (MTSP). Given a set of n targets along with the distance between each pair of targets, the TSP requires us to visit each target exactly once and return to the starting point along the shortest path possible. Thus, in a single TSP, there will be one UAV which has to traverse n targets along the shortest path, whereas in an MTSP, there will be m UAVs traversing n targets with the objective to complete the mission in minimum time. The objective of this research is to propose a method of solving the MTSP which gives good results while also being computationally efficient, thus coming up with an effective path planning algorithm. For n targets, there are $\frac{(n-1)!}{2}$ possible solutions for the TSP. For large values of n, due to the factorial term, the number of possible solutions is very high. Hence, it is computationally challenging to find the exact solution for large problem sizes.

Mathematically, the importance of the TSP is that it is representative of a larger class of problems known as combinatorial optimization problems. The TSP is a Non-deterministic Polynomial-time hard (NP-hard) problem, which means that the TSP belongs to a class of problems which are at least as hard as the hardest problems in NP. Hence, if one can find an efficient algorithm for the TSP, then efficient algorithms could be found for all other problems in NP. In complexity theory, the main question is whether or not there exists a polynomial time algorithm for an NP-hard problem.

In this paper, three different techniques to solve the TSP, viz. 2-opt method, Genetic Algorithm (GA) [5] and Particle Swarm Optimization (PSO) [6] are discussed. Furthermore, two of these techniques, the 2-opt and the GA, are used to solve the MTSP using a cluster first approach, which is compared to the conventional approach of applying GA directly to find the solution. The mission time is proportional to the minimum of the maximum distance amongst the UAVs. Hence, this is a min-max optimization problem.

There is plenty of research that has been done in the field of path planning which encompasses the TSP and the MTSP. Different approaches for solving the TSP and MTSP have been studied. One of the major limiting factors when it comes to solving TSP is the scalability of the technique; i.e., how well it performs as the number of targets, n, increases. With most algorithms, the computational time increases exponentially as n increases. Lin and Kernighan [7] came up with a heuristic method that produced near-optimum solutions with computational times proportional to n^2. This procedure is based on a general approach and is currently used successfully in solving a wide range of problems. GA has been used extensively to solve the TSP. Albayrak and Allahverd [8] developed a new mutation operator to increase the performance of GA in solving TSP. Chen and Chien [9] developed a method called the parallelized genetic ant colony system (PGACS), for solving the TSP which essentially consists of the GA with modified crossover and the hybrid mutation operations, done in parallel with ant colony systems. Soler, Yuichi and Nagata [10] used an edge-assembly crossover (EAX) operator [11], where offspring solutions are obtained by combining edges or arcs from two parent solutions and adding relatively few short edges or arcs. PSO is a very efficient technique for solving continuous problems. TSP being a discrete optimization problem has found very little use out of PSO. Ponnambalam *et a*l. [6] applied discrete PSO for flow-shop scheduling which refers to allocation of resources over time to perform a set of operations. Wang *et al*. [12] modified the discrete PSO by making use of the swap operator to construct the path of TSP. Pang *et al*. [13] combined fuzzy logic with PSO for solving the TSP. Fuzzy matrices were used to represent the position and velocity of the particles in PSO and the operators in the original PSO formulas were redefined. Liao *et al*. [14] developed an improved genetic-based PSO for solving TSP, which consisted of two phases. The first phase includes fuzzy c-means clustering, a rule-based route permutation, a random swap strategy and a cluster merge procedure. The genetic-based PSO procedure is then applied to solve the TSP with better efficiency in the second phase. Mitchell *et al*. [15] used fuzzy optimization of a path, produced using GA, to solve the TSP. The project was made more realistic by using Dubins paths and finally comparing this to genetic fuzzy algorithm to determine differences in accuracy and precision between the two solutions.

Golden *et al*. [16] proposed a heuristic approach for the class of Vehicle Routing Problems (VRP), which is performed by doing a tabu-search and an adaptive memory procedure. The VRP is a variant of the MTSP. Chistofides *et al*. [17] presented tree search algorithms for solving the VRP by incorporating lower bounds computed from shortest spanning *k*-degree centre tree (*k*-DCT), and *q*-routes. The results show that the bounds derived from the *q*-routes are superior to those from *k*-DCT and that VRPs of up to about 25 customers can be solved exactly. Kivelevitch *et al*. [18] proposed a method by simulating an economic market in which the agents (UAVs) interact to win tasks situated in an environment. The agents strive to minimize required costs, defined as either the total distance travelled by all agents or the maximum distance travelled by any agent. The paper shows that the MBS is both quick and close to the optimal solution, and robust to changes in the scenario (e.g., the addition and removal of tasks or agents). They discussed the problem of scalability [19]; i.e., how well the MBS performs when applied to larger sized problems, in for a min-max variant of Multiple Depot MTSP (MDMTSP). Carlsson *et al.* [20] explored the min-max problem using two heuristic methods. The first approach is based on linear programming with global improvement and the other is a region partition heuristic, which they proved to be asymptotically optimal and can be applied for general network applications. Campbell *et al*. [21] presents approaches for solving the VRP based on well-known insertion and local search techniques that are used in a series of computational experiments to help identify the instances in which TSP and VRP solutions can be significantly different from optimal min-max solutions.

Ernest and Cohen [1] developed a GA and Fuzzy Logic System (FLS) based approach to the path representation of a variant of the TSP known as the Multi-Depot Polygon Visiting Dubins Multiple Travelling Salesman Problem (MDPVDMTSP). Utilizing a hybridization of control techniques, they proved that this approach works effectively and efficiently approximates path planning and visibility problems encountered by a UAV swarm in a constant altitude, constant velocity, two-dimensional case. Ernest *et al.* [22] provided approximate solutions for complex variants of the TSP, or more precisely named the Multi-Objective Min-Max Multi-Depot Polygon Visiting Dubins Multiple Travelling Salesman Problem (MMMPVDMTSP). The techniques are utilized in such a way that the problem is examined from a top level view which is then approximated entirely before moving on to the next level. While iterative methods are used at almost every level of the problem, each level is only solved once. Assumptions and generalizations must be made to accommodate this, however the cost of these can be minimized and the payoff is drastically reduced runtime, even for such a complex problem.

II. Mathematical Formulation

2.1 Single TSP

Let $N = \{1, 2, 3, \ldots, n\text{-}1, n\}$ be the set of indices defining the targets and d_{ij} be the distance between the i^{th} and j^{th} targets. The problem is assumed to be symmetric so that:

$$d_{ij} = d_{ji} \;\forall\; i, j \;\in N \tag{1}$$

It is assumed that the triangle inequality holds true, i.e.

$$d_{ij} + d_{jk} \geq d_{ik} \;\forall\; i, j, k \;\in N \tag{2}$$

The TSP can be formulated as an integer linear problem. Let T be a vector that represents the tour.

$$T = [a_1 a_2\ a_3 \ldots \ldots a_n\ a_1] \quad (3)$$

$$a_1, a_2, a_3, \ldots \ldots, a_n \in N \quad (4)$$

For T to be a tour satisfying the conditions of TSP, T should be of length $n+1$ with each value in T being unique.

$$a_1 \neq a_2 \neq a_3 \neq \cdots \ldots . \neq a_n \quad (5)$$

$$\dim(T) = n \quad (6)$$

The total distance traversed by the UAV following tour T is given by,

$$D = \sum_{p=1}^{n} d_{a_p a_{p+1}} \quad (7)$$

a_p and a_{p+1} are indices of the targets. The total distance traversed by the UAV, given by D, is the cost function that needs to be minimized. Thus, as the objective is to

$$minimize \quad D = \sum_{p=1}^{n} d_{a_p a_{p+1}} \quad (8)$$

2.2 Multiple TSP

Let there be m UAVs starting off from a single depot and hence there will be m tours T_1, T_2, T_3, … , T_m. The index of the depot is $n+1$. The tour traversed by the q^{th} UAV, assuming it has n_q targets, and the corresponding distance is given by

$$T_q = \left[n+1\ qa_1\ qa_2 \ldots \ldots qa_{n_q}\ n+1\right];\ q = 1, 2, 3, \ldots \ldots, m. \quad (9)$$

$$D_q = d_{n+1, qa_1} + \sum_{p=1}^{n_q} d_{a_p a_{p+1}} + d_{qa_{n_q}, n+1} \quad (10)$$

The number of targets covered by each UAV added together should be equal to the total number of targets on the map.

$$\sum_{q=1}^{m} n_q = n \quad (11)$$

In the MTSP, the objective is to minimize the total mission time t_M.

$$minimize \qquad t_M \quad (12)$$

The mission time is equal to the time taken by the last UAV to get back to the depot, which is proportional to the distance travelled by this UAV, since the speed of all the UAVs are assumed to be equal. Thus, the mission time is proportional to the maximum distance travelled amongst the m UAVs, D_{max}.

$$D_{max} = \max_q (D_q) \quad (13)$$

Hence, the objective can be rewritten as

$$minimize \qquad D_{max} = \max_q (D_q) \quad (14)$$

From *eq.* 14, it is clear why the MTSP is called the min-max optimization problem.

III. TECHNIQUES TO SOLVE TSP

3.1 2-opt method

The 2-opt method is a simplified form of the Lin-Kernighan algorithm [8], also known as k-opt. This was developed by Shen Lin & Brian Kernighan. It is basically a tour improvement method which takes a given tour and attempts to modify it in order to obtain an alternative tour of lower cost. A good tour can be obtained by repeatedly replacing sets of tour edges by cheaper alternative sets wherever possible. The flowchart of 2-opt method is shown in Fig. 1. As shown in the flowchart, if *eq.* 15 is satisfied, then the indices of targets $i+1$ and j should be swapped in the tour T.

$$E_{i,i+1} + E_{j,j+1} > E_{i,j} + E_{i+1,j+1} \tag{15}$$

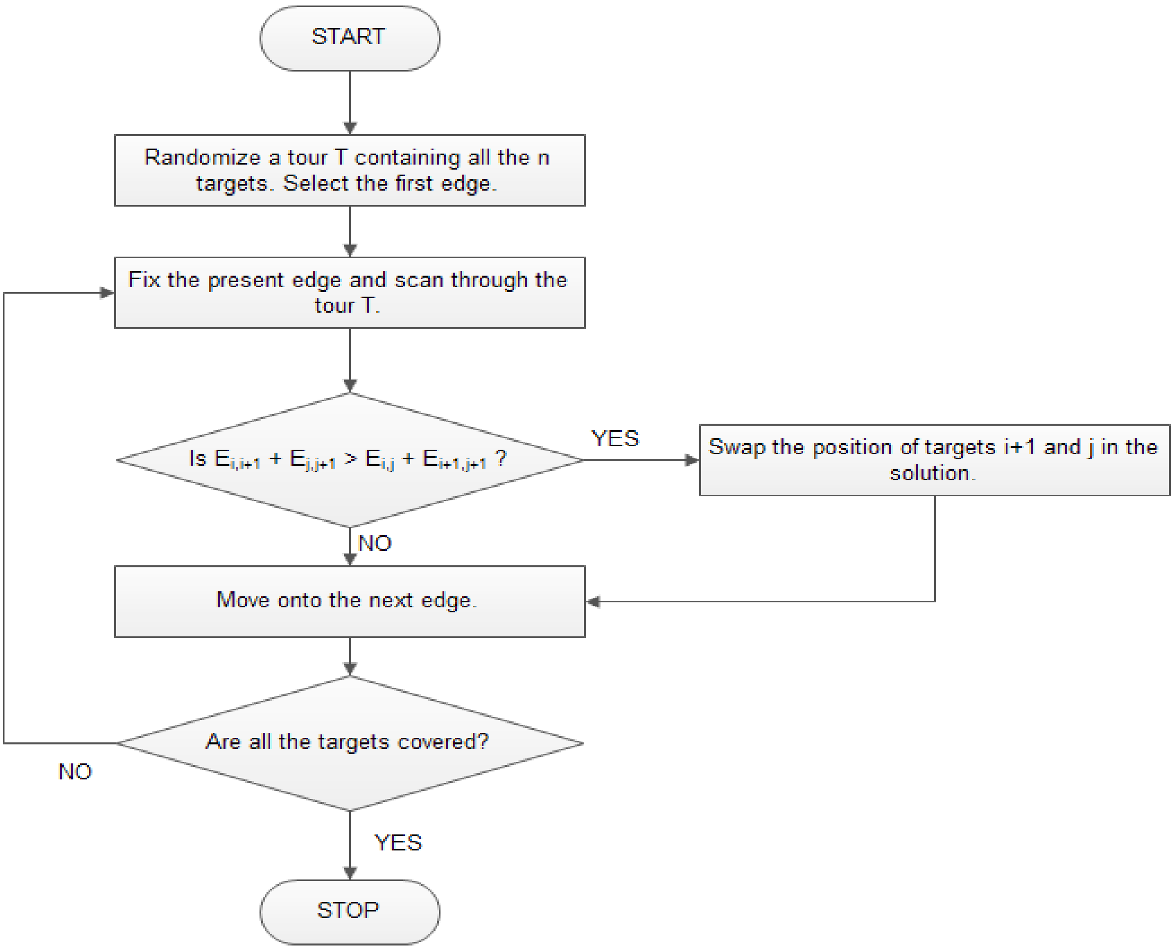

Fig. 1: Flowchart: 2-opt

The pseudo-code for the 2-opt method is given in Table 1. Instead of starting from a random tour, the algorithm could as well be initialized with the tour obtained by applying the nearest neighbour (NN) approach. In the nearest neighbour algorithm, the salesman starts at a random target and repeatedly visits the nearest target until all the targets have been covered. It is shown in Section 4 that this approach gives a slightly improved computational efficiency.

Table 1: Pseudo-code for 2-opt algorithm

Pseudo-code: 2-opt algorithm

Require: *n targets and its xy locations.*

```
1: evaluate the distance matrix.
2: define a tour T initialised randomly or the tour obtained by using Nearest Neighbour approach.
3: for i = 1:n-2,
4:     for j = i+2:n,
5:         evaluate d1 = total length of the 2 edges.
6:         evaluate d2 = total length of the edges when the targets are swapped.
7:         if d1 > d2
8:             Swap indices of targets in tour T.
9:         else
10:        end
11:    end
12: end
```

3.2 Genetic Algorithm (GA)

GA is based on the mechanics of biological evolution and is one of the algorithms that have found extensive use in solving optimization problems. The flowchart and pseudocode for GA are shown in Fig. 2 and Table 2, respectively.

Table 2: Pseudo-code - Genetic Algorithm (GA)

Pseudo-code: Genetic Algorithm (GA)

Require: *n targets and its xy locations.*

```
1: evaluate the distance matrix.
2: create a population and define the crossover and mutation functions and their
   probabilities.
3: while (stall generation limit NOT reached)
4:     evaluate the fitness function of each chromosome.
5:     select few of the best chromosomes.
6:     perform crossover operations on pairs of selected chromosomes.
7:     perform mutation based on mutation probability
8: end
```

3.3 Particle Swarm Optimization

The PSO is originally attributed to James Kennedy and Russell Eberhart [23] and was intended to simulate social behavior involved in the movement of flock of birds. The flowchart for the PSO algorithm is shown in Fig. 3.

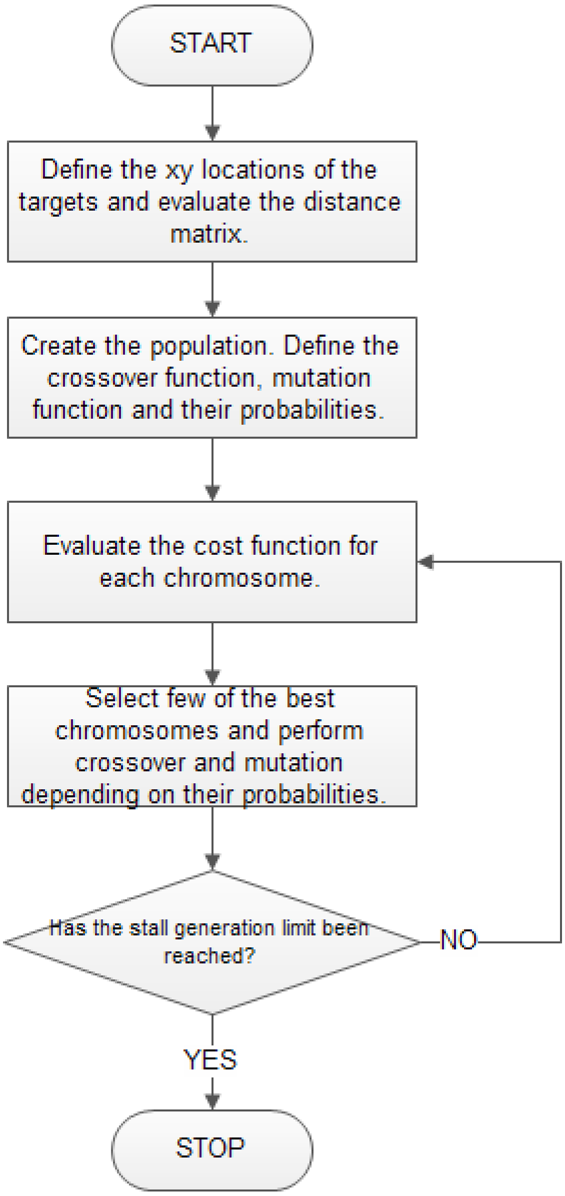

Fig. 2: Flowchart: Genetic Algorithm

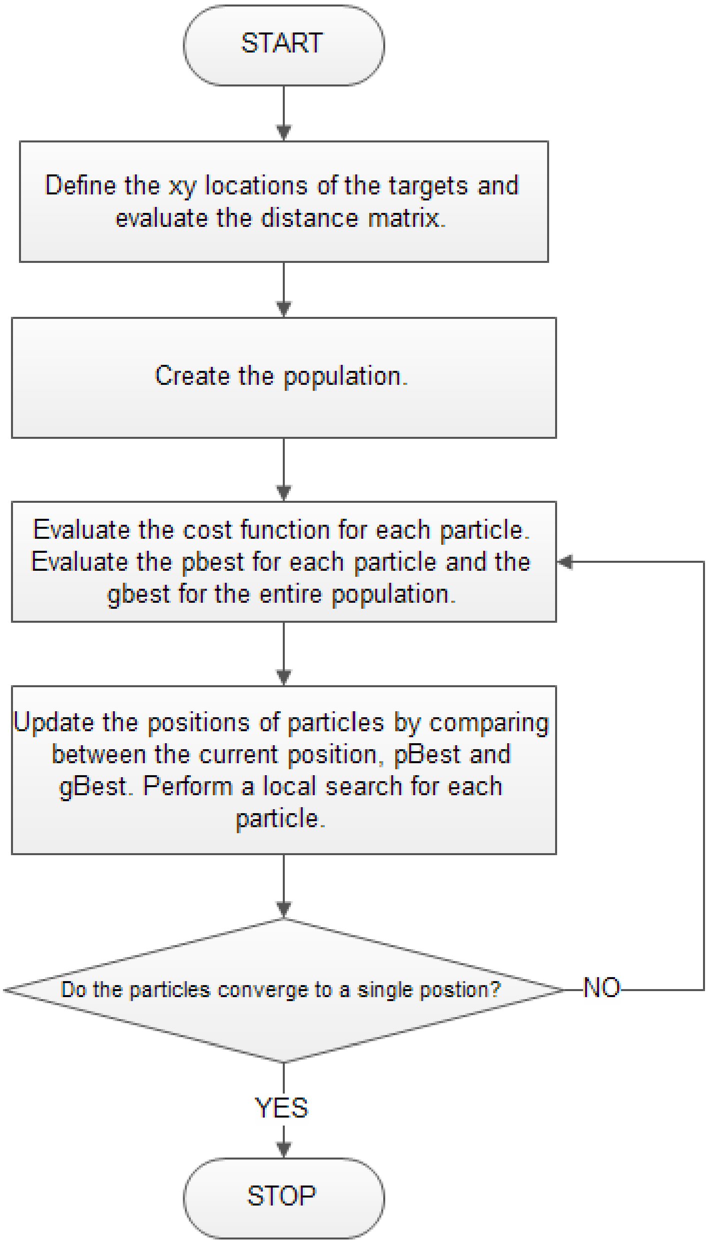

Fig. 3: Flowchart: Particle Swarm Optimization (PSO)

The algorithm is initialized by creating a swarm of random solutions in the search space, known as particles, which together constitute the population. Each particle knows its position and velocity, as well as its personal best position (pbest) and global best position (gbest). The pbest for each particle is the location where that particular particle had the best solution and gbest is the global best among all the pbest locations. The behavior of each particle is a compromise between three possible choices: to follow its own way, to move towards pbest, or to move towards gbest. Since the TSP is a discrete optimization problem, the discrete form of the PSO [7] is used. The pseudo-code of PSO is given in Table 3.

Table 3: Pseudo-code - Particle Swarm Optimization (PSO)

Pseudo-code: Particle Swarm Optimization (PSO)

Require: *n targets and its xy locations.*
1: evaluate the distance matrix.
2: create a population.
3: while (solution NOT converged)
4: evaluate the fitness function associated with each particle.
5: evaluate pBest for each particle and gBest associated with entire population.
6: update the positions of particles by converting the particle in such a way that it becomes more similar to pBest and gBest values.
7: perform local search for each particle.
8: end

3.4 K-means clustering

K-means clustering is the process of partitioning n points into k sets on the basis of their spatial distribution [24]. The flowchart for the K-means clustering algorithm is given in Fig. 4. The main idea is to define k centroids, one for each cluster, in such a way that they are placed as far away from each other as possible [25]. The next step is to take each point in the data set and associate it to the nearest centroid. Then, re-calculate k new centroids for each of the clusters resulting from the previous step. This is repeated until no more changes are made. For an MTSP, n targets can be clustered into k different sets and then one of the three algorithms discussed before could be used to solve the individual TSPs.

IV. Results and Discussion

4.1 Single TSP

In this section, a comparison of the results obtained by solving the single TSP using each of the three algorithms is described. A laptop with an Intel Core i3, 2.3 GHz processor, 4 GB RAM was used. The problem was simulated in MATLAB. For the purpose of comparison, a scenario involving 50 targets randomly placed over a 1 unit x 1 unit map is considered for each of the three algorithms. The TSP is solved for 100 different distributions of the targets using the three algorithms. The minimum distances for each of these 100 different distributions and their computational times are compared in Table 4. The values are normalized with respect to the corresponding mean values obtained for GA. As shown in Table 4, an improvement in computational time is obtained when the 2-opt algorithm is initiated using the result obtained using nearest neighbour algorithm instead of a random choice.

Table 4: Comparison of the three algorithms for a single TSP

	Average of Min. Distance	Average Computational Time	Variance in Distances	Minimum Value	Maximum Value
2-opt	1.023	0.0051	0.122	0.891	1.152
2-opt with NN	1.023	0.0044	0.122	0.892	1.152
PSO	1.080	2.7900	0.043	1.024	1.149
GA	1.000	1.0000	0.100	0.848	1.124

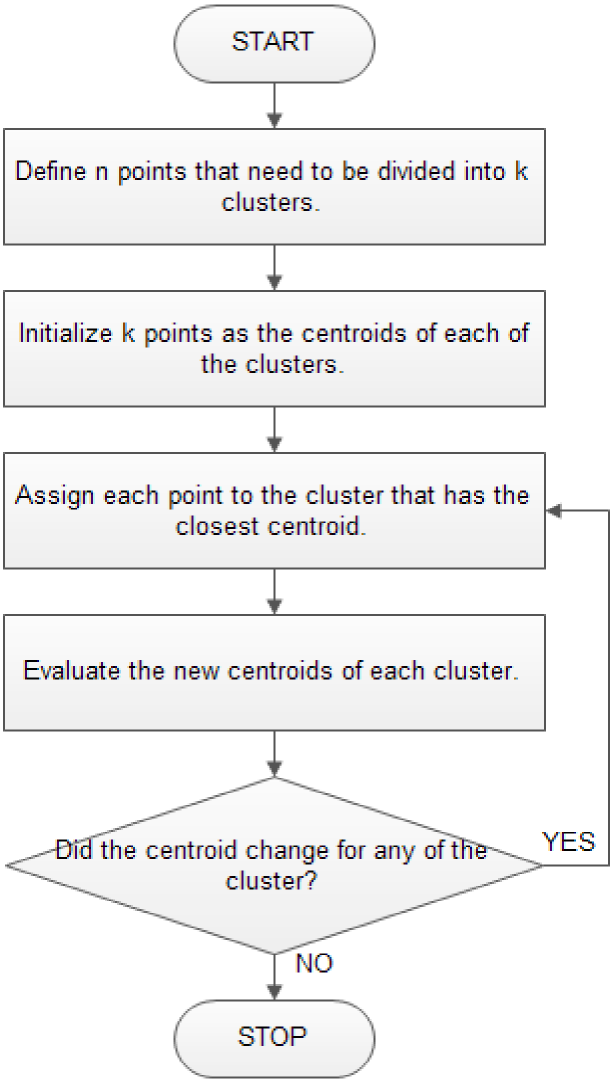

Fig. 4: Flowchart: K-means algorithm

For a 50-target TSP, the number of possible routes is $\frac{49!}{2} = 3.04 \times 10^{62}$. This is a very large solution space. Both GA as well as PSO have a systematic approach to traversing such a huge search space, whereas the 2-opt modifies an existing tour to obtain better tours. As Table 4 shows, the 2-opt method is the most computationally fast and efficient technique followed by GA. GA is an excellent tool for optimization. By tuning a few parameters like population size, generation, stall generation limit and mutation rate, one is able to obtain excellent results to even a difficult optimization problem like the TSP. PSO is the more consistent as shown by its low variance although it takes longer computational times and also the minimum obtained using PSO is higher compared to the other two. GA gives good solutions although its computational time is higher compared to 2-opt. Thus, for applications that require fast computation at the cost of slight loss in performance, 2-opt is the best approach to use.

4.2 Multiple TSP

In the MTSP, the objective is to cover a certain number of targets using a given number of UAVs in the minimum time. A scenario containing four UAVs to cover 200 targets spatially distributed on a 10 units x 10 units map is considered. All the four UAVs start from a common depot and shall return once all the targets have been covered. The depot is located at the center of the map with coordinates (5, 5). The location of the depot does not affect the performance of the algorithm. The objective of a UAV swarming problems is to reduce mission time. Mission time is equal to the maximum time taken amongst the four UAVs. In order to minimize mission time, the maximum distance covered amongst the UAVs needs to be minimized. Thus, an MTSP is a min-max optimization problem.

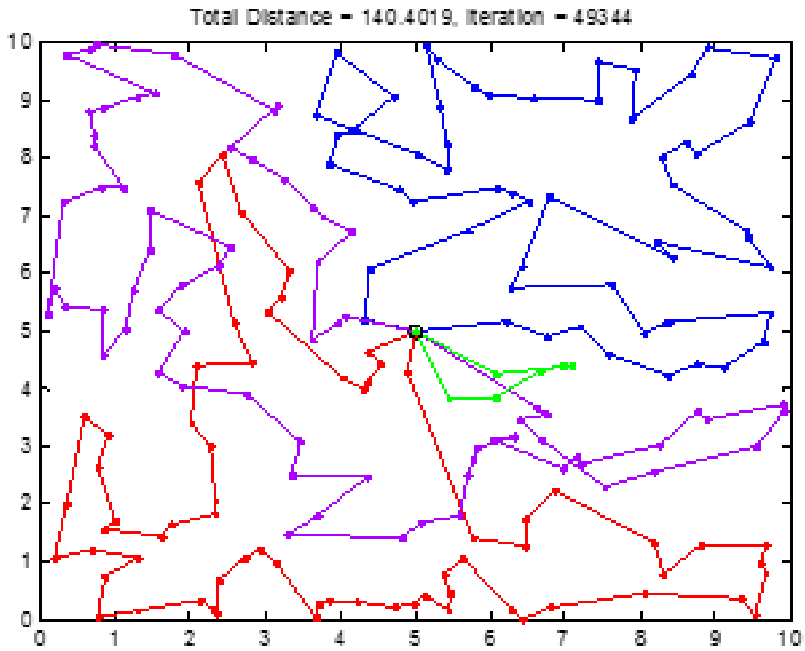

Fig. 5: MTSP using GA

A common approach to solving an MTSP is to apply GA directly, with the cost function being the maximum distance among the four UAVs. The result obtained by directly applying GA is shown in Fig. 5. Different colored paths indicate the paths for different UAVs. As can be seen from Fig. 5, the green colored UAV travels a very small distance while the others have to compensate for that. Thus, this is not an optimum distribution of targets amongst the four UAVs and hence not an optimum solution.

In this study, k-means is used to cluster the targets into four sections and then GA or 2-opt method is used to solve the individual TSP for each cluster. The results obtained using GA and 2-opt method are shown in Figs. 6 and 7, respectively. It is obvious from Figs. 6 and 7 that the cluster-first approach gives much better solutions than the GA shown in Fig. 5. The computation for 20 random positions of these 200 targets was done. The results obtained are tabulated in Table 5. The values are normalized with respect to the corresponding mean values obtained by directly applying GA.

As can be seen from the results shown in Table 5, the cluster-first approach is far more efficient than directly applying GA to the MTSP. 2-opt gives better results compared to GA and it fits perfectly well for the MTSP. The cluster-first approach has decreased the computational time by 93% in case of GA and 99.85% for 2-opt using NN algorithm. Since 2-opt is fast, it can be used for real time updates during the mission. For example, if one of the UAVs is shot down or if a new set of targets are added to the existing mission, this algorithm can help to rearrange the UAVs to cover the target areas.

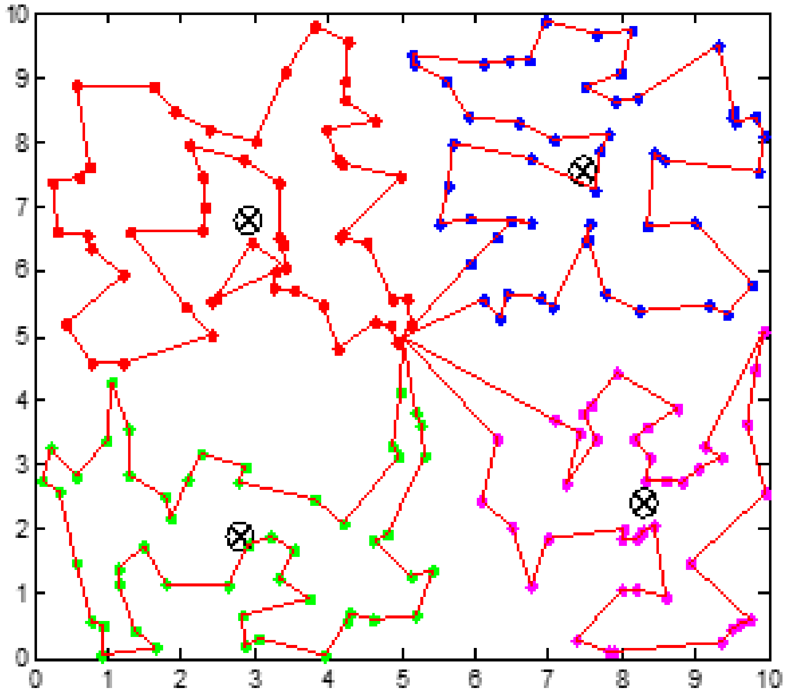

Fig. 6: Cluster first GA

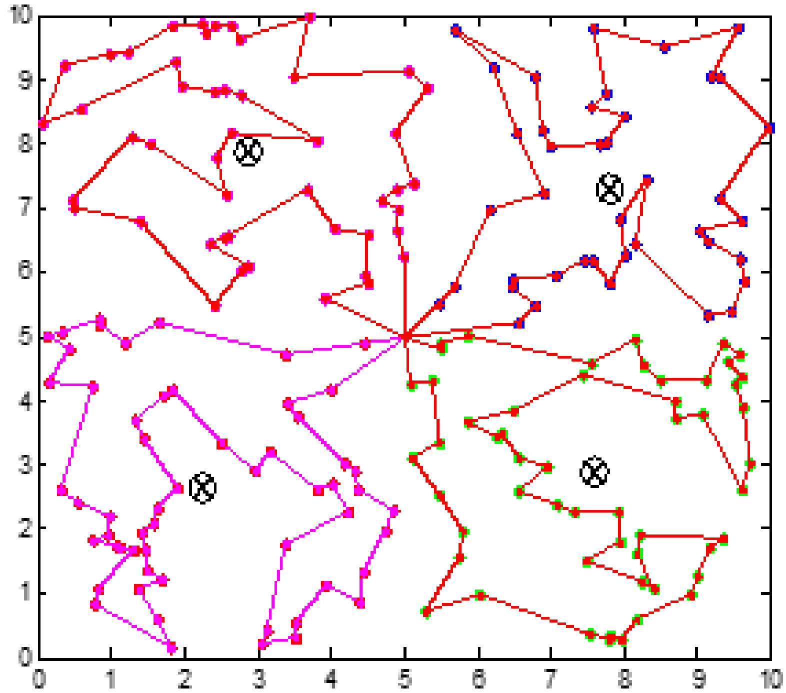

Fig. 7: Cluster first 2-opt

Table 5: MTSP Results

	Average of Largest Distance	Average Computational Time	Variance in Distances
GA	1.0000	1.0000	0.241
Cluster-GA	0.6976	0.0792	0.128
Cluster-2opt	0.6817	0.0017	0.104
Cluster-2opt with NN	0.6817	0.0015	0.104

4.3 Scalability

It has already been shown that the cluster-first approach improves the computational efficiency by more than 93% irrespective of whether 2-opt or GA is used to solve the individual clusters. However, one important part of the analysis involves how well the algorithm scales; i.e., how does increasing the problem size, viz. the number of targets and number of UAVs, affect the performance of the algorithm. In order to analyze the scalability, cluster-first approach along with 2-opt was used to obtain the computational times for various scenarios in which the number of targets were varied from 70 to 150 at intervals of 10 and the number of UAVs were varied from 1 to 7. The results obtained are shown in Fig. 8.

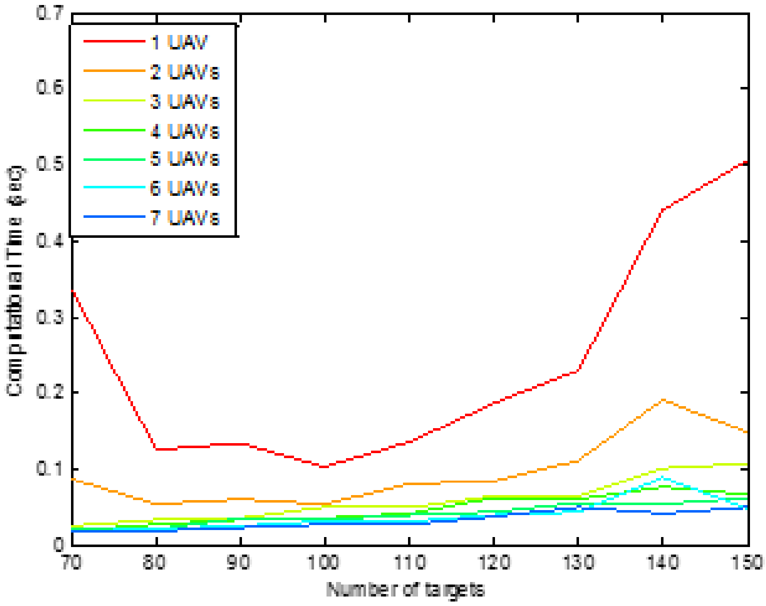

Fig. 8: Computational times as problem size is increased

As can be seen from Fig. 8, the computational times decrease as the number of targets is increased. This is mainly because of the cluster-first approach in which the whole target space is divided into different clusters and then solved each cluster. For one set of UAVs, the computational time plot is seen to be approximately flat as the number of targets increases. This is a very interesting result since the TSP being an NP-hard problem is expected to have exponential increase in computational time as the problem sizes become larger. However, instead it can be observed that the plots are essentially flat as the number of targets increase and the computational times decrease as more UAVs are added. Hence, it can be concluded that this algorithm is effective and scalable to large problem sizes.

V. Conclusions and Future Work

In this paper, two important optimization problems, the TSP and the MTSP, and how they are applied for UAV swarming, were discussed. As previously described the TSP and its variant, the MTSP, finds variety of applications in engineering such as UAV swarming, drilling holes in PCB, order-picking problems in warehouses, and other. Thus, the TSP is an important research area and its solution helps in providing more efficient ways to perform these engineering applications. The mathematical formulations for the TSP and MTSP, as well as the three algorithms used to solve them viz. the 2-opt method, GA and PSO were discussed. The performance of the three algorithms in terms of the minimum distance obtained (cost function) as well as the computational times required were compared. It was found that the 2-opt method is the most computationally efficient although the minimum distance obtained was slightly greater compared to GA. The PSO took more time to compute and rarely came up with an optimal solution. In the case of MTSP, cluster-first approach is used followed by solving individual clusters using 2-opt as well as GA and compared them to existing method of using GA directly on the min-max optimization problem. The cluster-first approach decreased the computational time by 93% in case of GA and 99.85% for 2-opt using NN algorithm. Also, a 32% reduction in the largest distance amongst the four UAVs was obtained. Thus, overall it can be concluded that the cluster-first approach is definitely a great improvement over directly applying GA to solve the MTSP. Also, the cluster-first 2-opt gives better results, both in terms of computational time and the minimum cost function, as compared to cluster-first GA.

As a next step, we intend to apply genetic fuzzy logic for clustering in the MTSP and see how it improves the results. Prior works [1-3] in this regard show that the results obtained by following genetic fuzzy approach are superior. Also, we intend to increase the complexity of the MTSP by considering targets as polygons instead of points so that the UAV has to just touch the polygons as opposed to travelling to its center. This would decrease the total distance covered by the UAVs but will be computationally more complex to solve.

VI. References

1. **Ernest N** and **Cohen K**. (2012). Fuzzy clustering based genetic algorithm for the multi-depot polygon visiting dubins multiple traveling salesman problem. *Proceedings of the 2012 IAAA Infotech@ Aerospace.* No. AIAA-2012-2562. AIAA. Garden Grove, CA. crossref
2. **Ernest N, Cohen K** and **Schumacher C**. (2013). Collaborative tasking of UAV's using a genetic fuzzy approach. *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition.* 7th – 10th January. Grapevine (Dallas/Ft. Worth Region), Texas. crossref
3. **Park TH** and **Kim HJ**. (2005). Path planning of automatic optical inspection machines for PCB assembly systems. *Computational Intelligence in Robotics and Automation, 2005. IEEE International Symposium.* Pp. 249-254. crossref
4. **Matai R, Singh SP** and **Mittal ML**. (2010). Traveling salesman problem: An overview of applications, formulations, and solution approaches. Traveling Salesman Problem, Theory and Applications. *InTech.* Pp. 1-24. crossref crossref
5. **Mitchell M**. (1998). *An introduction to genetic algorithms*. MIT press. Cambridge, MA.
6. **Ponnambalam SG, Jawahar N** and **Chandrasekaran S**. (2009). Discrete particle swarm optimization algorithm for flowshop scheduling. Particle Swarm Optimization. *InTech.* Pp. 397-422. crossref
7. **Lin S** and **Kernighan BW**. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*. ***21***(2): 498-516. crossref
8. **Albayrak M** and **Allahverdi N**. (2011). Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms. *Expert Systems with Applications*. ***38***(3): 1313-1320. crossref
9. **Chen SM** and **Chien CY**. (2011). Parallelized genetic ant colony systems for solving the traveling salesman problem. *Expert Systems with Applications*. ***38***(4): 3873-3883.crossref
10. **Nagata Y** and **Soler D**. (2012). A new genetic algorithm for the asymmetric travelling

salesman problem. *Expert Systems with Applications*. ***39***(10): 8947-8953. crossref
11. **Nagata Y**. (1997). Edge Assembly Crossover. A high-power genetic algorithm for the traveling salesman problem. *Proceedings of the 7th ICGA*. 19th -23rd July. East Lansing, MI. Pp. 450-457. crossref
12. **Wang KP, Huang L, Zhou CG** and **Pang W**. (2003). Particle swarm optimization for traveling salesman problem. *Machine Learning and Cybernetics*. ***3***: 1583-1585. crossref
13. **Pang W, Wang KP, Zhou CG** and **Dong LJ**. (2004). Fuzzy discrete particle swarm optimization for solving traveling salesman problem. *Computer and Information Technology*. Pp. 796-800. crossref
14. **Liao YF, Yau DH** and **Chen CL**. (2012). Evolutionary algorithm to traveling salesman problems. *Computers & Mathematics with Applications*. ***64***(5): 788-797. crossref
15. **Mitchell SM, Ernest ND** and **Cohen K**. (2013). Comparison of Fuzzy Optimization and Genetic Fuzzy methods in solving a Modified Traveling Salesman Problem. *AIAA Infotech@Aerospace Conference.* Paper 2013-4664. At Boston, MA. crossref
16. **Golden BL, Laporte G** and **Taillard ÉD**. (1997). An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. *Computers & Operations Research*. ***24***(5): 445-452. crossref
17. **Christofides N, Mingozzi A** and **Toth P**. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*. ***20***(1): 255-282. crossref
18. **Kivelevitch E, Cohen K** and **Kumar M**. (2011). Market-based solution to the allocation of tasks to agents. *Procedia Computer Science*. ***6***: 28-33. crossref
19. **Kivelevitch E, Cohen K** and **Kumar M**. (2012). On the scalability of the market-based solution to the multiple traveling salesmen problem. *AIAA Infotech@Aerospace Conference.* 19th – 21st June. Garden Grove. California. crossref
20. **Carlsson J, Ge D, Subramaniam A, Wu A** and **Ye Y**. (2009). Solving min-max multi-depot vehicle routing problem. Lectures on Global Optimization. *Fields Institute Communications*. ***55***: 31-46. crossref
21. **Campbell AM, Vandenbussche D** and **Hermann W**. (2008). Routing for relief efforts. *Transportation Science*. ***42***(2): 127-145. crossref
22. **Ernest N, Cohen K** and **Schumacher C**. (2013). UAV swarm routing through Genetic Fuzzy Learning methods. *AIAA Infotech@Aerospace Conference.* 19th - 22nd August. Boston, Massachusetts. USA. Pp. 624-632. crossref
23. **Kennedy J**. (1997). The particle swarm: social adaptation of knowledge. *Evolutionary Computation.* Pp. 303-308. crossref
24. **MacQueen J**. (1967). Some methods for classification and analysis of multivariate observations. *5th Berkeley Symposium on Mathematical Statistics and Probability*. ***1***(14): 281-297. crossref
25. **Politecnico di Milano.** (2014). http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html Accessed: May 2014. crossref

7. Notation

D	Total distance travelled by UAV following tour T, which is also the cost function in the case of single TSP
d_{ij}	Distance between i^{th} and j^{th} targets
D_{max}	Maximum distance covered amongst the m UAVs
D_q	Distance traversed by q^{th} UAV
$E_{i,j}$	Length of the edge connecting the i^{th} and j^{th} targets in the tour T
f_i	Fitness of the i^{th} chromosome in GA
m	Number of UAVs
M	Number of chromosomes in the population
n	Number of targets
N	Set of indices defining the targets

n_q	Number of targets covered by q^{th} UAV
p_i	Probability of selecting the i^{th} chromosome for crossover
q	Index representing the UAV
T	Vector representing the tour
t_M	Total mission time in the case of MTSP