ISSN: 0976-3104
SPECIAL ISSUE: Computer Science

ARTICLE

A SURVEY ON MULTI-OBJECTIVE TRAVELLING SALSMAN PROBLEM

Kanimozhi Jayamoorthi[1*], Dinesh Karunanidy[2], Amudhavel Jayavel[2], Subramanian Ramalingam[1]

[1]*Department of Computer Science and Engineering, Pondicherry University, Puducherry, INDIA*
[2]*Department of Computer Science and Engineering, KL University, Andhra Pradesh, INDIA*

ABSTRACT

Traveling Travelling Salesman Problem (TSP) is a challenging problem in combinatorial optimization. The important of this problem is due to the fact that it is used in many fields such as transportation, logistics, semiconductor industry, problem of routing, scan chain optimization and drilling problem in integrated orbit test, production and many others scientific and industrial fields. In this paper we consider the multi-objective TSP (MOTSP) which is an extended instance of traveling salesman problem (TSP). Since TSP is a NP- hard problem MOTSP is also NP-hard problem. In MOTSP simultaneous optimization of more than one objective functions is required. TSP considers and optimizes one objective function to find the best solution. Instead in MOTSP many objectives are considered and optimized to find the best solutions. Many algorithms are used to solve MOTSP. Some algorithms give optimal solution and some give the nearest optimal solution. By evolving a population of solutions, multi-objective evolutionary algorithms (MOEAs) are able to approximate the Pareto optimal set in a single run. It results in nearest optimal solution within a reasonable amount of time by optimizing many objectives. In this survey, we review the current state-of-the-art computational algorithms used to solve the MOTSP.

KEY WORDS
Multi-objective travelling salesman problem, Multi-objective evolutionary algorithm, NSGA II, Decomposition, Ant colony optimization

Received: 3 June 2017
Accepted: 20 July 2017
Published: 15 Sept 2017

Corresponding Author
Email:
janathakani@gmail.com
Tel.: +91 7373231828

INTRODUCTION

The Travelling Salesman Problem (TSP) is an optimization problem used to find the shortest path to travel through the given number of cities. Travelling salesman problem states that given a number of cities N and the distance or time to travel between the cities, the traveler has to travel through all the given cities exactly once and return to the same city from where he started and also the cost of the path is minimized. This path is called as the tour and the path length or travel time is the cost of the path [1] [2]. The TSP mathematical model follows [3] [2]:

$$\min Z = \sum_{i=1}^{N}\sum_{j=1}^{N} X_{ij}C_{ij}$$

Subject to $\sum_{i=1}^{N} X_{ij} = 1$ j=1,2,3.....N

$\sum_{j=1}^{N} X_{ij} = 1$ j=1,2,3..... N

Let C_{ij} be the cost for traveling from i-th city to j-th city. Where Xij = 1 if the salesman travels from city-i to city-j, otherwise Xij = 0. In the mathematical model it is shown that the distance is considered for finding the best solution.

The travelling salesman problem can be classified as symmetric Travelling Salesman Problem (STSP), Asymmetric Travelling Salesman Problem (ATSP), and Multi Travelling Salesman Problem (MTSP) [9] [1]. In travelling salesman problem with increasing the number of cities the existing solutions don't provide optimal solution at the appropriate time. Moreover the solution found by optimizing the distance to travel to the destination may not always give the best solutions. For the solution optimization more than one objective needs to be considered. When more objectives functions are considered to optimize the solutions then it will give the better solutions compared to solutions given by single objective function [5]. This gives the need for MOTSP. MOTSP considers more than one objective functions to optimize the solutions. In MOTSP, the aim is to simultaneously optimize several conflicting objectives, such as shortest travelling distance, minimum time, minimum cost, and lowest risk. Under multi-objective framework no single point is considered as an optimal solution, because an improvement in one objective will cause at least another objective not being able to be optimized. Therefore, the optimal solution can only be a set of non-dominated and trade-off solutions. The [6] gives the mathematical model of MOTSP by considering two objectives for optimization.

The MOTSP can be formulated as a multi-objective model with two objective functions. The first objective function (1) considers the minimization of the distance traveled by the salesman, while the second objective function (2) considers the working time of the salesman.

$$(1) \quad \min Z1 = \sum_{i=1}^{N}\sum_{j=1}^{N} X_{ij}C_{ij}$$

ISSN: 0976-3104

$$(2) \quad \min Z2 = \sum_{i=1}^{N}\sum_{j=1}^{N} X_{ij}T_{ij}$$

Subject to $\sum_{i=1}^{N} X_{ij} = 1$ j=1,2,3.....N

$\sum_{j=1}^{N} X_{ij} = 1$ j=1,2,3..... N

Let C_{ij} be the cost for traveling from i-th city to j-th city. T_{ij} is the travel cost from i-th city to j-th city and X_{ij} = 1 if the salesman travels from city-i to city-j, otherwise X_{ij} = 0. Both the objectives are optimized to find the best solutions.

MOTSP can be solved using conventional technique and Evolutionary based technique. [Fig. 1] classifies the techniques to solve MOTSP. Under conventional technique there are two ways to solve it. 1) Weighted Sum Technique: This technique converts multiple objectives into single objective using linear combination of objectives. But it requires a prior knowledge of weightage of each objective of a problem. 2) Constraint Based Technique: This technique considers only one objective at a time and treats remaining k-1 objectives as constraints. Final answer is computed by taking average of results obtained for all objectives. Application of this technique demands a prior knowledge of constraints of the problem. Due to these reasons both are not the best techniques to solve the MOTSP.

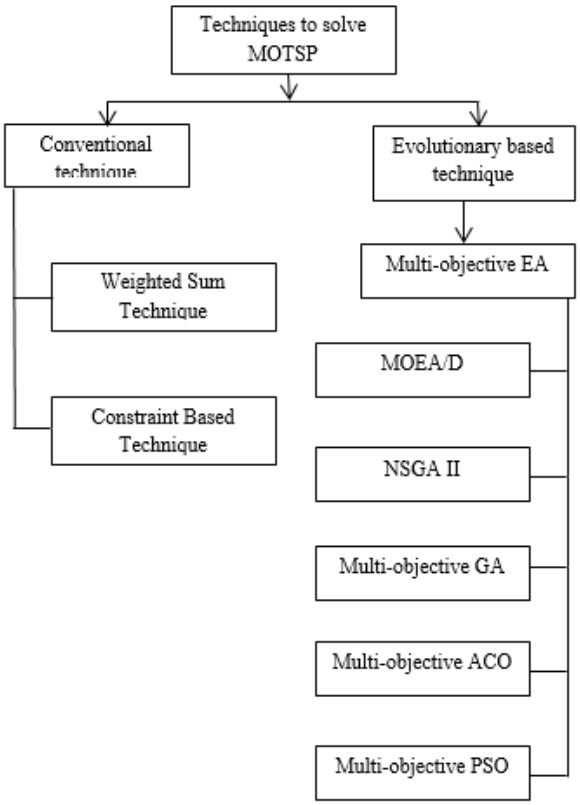

Fig. 1: Techniques to solve MOTSP

Under evolutionary based technique Multi-objective evolutionary algorithms (MOEAs) are well-suited for solving several complex multi-objective problems with more objectives. MOEA generates a set of non-dominated solutions at the end of each run. MOEA generates a set of non-dominated solutions at the end of each run, which is called Pareto set. The Pareto front contains set of Pareto solutions. Generally, an external archive is used by MOEAs to maintain a set of non-dominated Pareto set solutions [7] [4]. MOEA can also be used to solve single objective problems with or without constrained [67]. In addition to that bi-objective problems are solved using MOEA in [9] [10] [11] [12]. In [13] given the reasons to use MOEAs for solving multi-objective optimization problems :(i) they are easy to implement, (ii) MOEAs return more than one optimal solution, (iii) there are less chances of the algorithm to get stuck in local minima, (iv) MOEAs are flexible and robust, (v) MOEAs do not require any a prior knowledge of the problem.

As in [13] MOEAs can be classified into elitist and non-elitist algorithms. Elitist MOEAs have a mechanism to preserve good solutions at every generation while non-elitist MOEAs do not have such mechanism. Therefore, non-elitist algorithms perform worst monotonically than elitist algorithms. In MOEAs, a solution x is called non-dominated solution if it is better in all objectives than the solution y or solution x is strictly

IIOAB JOURNAL
ISSN: 0976-3104

better than the solution y in at least one objective. The solutions which do not satisfy above two conditions are called dominated solutions. In MOEAs, at the end of each generation, we have a set of non-dominated solutions. When we plot these non-dominated solutions on a graph, such graph is called Pareto front and Solutions on the Pareto front are called Pareto set (solutions). At the termination of MOEA, we have a set of non-dominated solutions. We obtained optimal Pareto front at the end of termination of MOEA and solutions on it are called optimal Pareto solutions.

MATERIALS AND METHODS

In this section we present a short overview of state-of-the-art computational algorithms as well as the working of those algorithms. Under Multiobjective Evolutionary Algorithm (EA) many algorithms are used to solve MOTSP. The following factors make the difference in solving the problem. 1) Expected fitness of the solution 2) The uncertainty of the solutions 3) Algorithm convergence speed 4) Usage of external archive. All these above factors give the different ways of solving the MOTSP. The categories of MOEA given by the above factors are follows:

State of art Computational Algorithms:

a. Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D)
b. Multi-objective Genetic Algorithm
c. NSGA II
d. Multi-objective Ant colony optimization
e. Multi-objective Particle swarm optimization

Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D)

MOEA/D [16] is a general EA framework for dealing with MOPs. Like generic MOEAs, a MOEA/D starts with an initial population of candidate solutions, in an iteration it generates some new trial solutions and selects the fittest ones to the next iteration; and it repeats the process until some termination conditions are satisfied. One of the major advantages of MOEA/D is that it is very easy to design local search operator within it using well-developed single-objective optimization algorithms. In MOEA/D, an MOTSP is decomposed into a set of scalar objective sub-problems and a probabilistic model, using both priori and learned information, is built to guide the search for each sub-problem. By the cooperation of neighbor sub-problems, MOEA/D could optimize all the sub-problems simultaneously and thus find an approximation to the original MOTSP in a single run [30] [31] [16]. This idea is realized by solution cooperation in neighborhood, i.e., the solutions in the same neighborhood are used to generate new trial solutions, and the new trial solutions only update the old solutions in the same neighborhood. The following two notations are extremely important in an MOEA/D [17].

Sub-problem: a multi-objective optimization problem is decomposed into a set of scalar objective problems and each of them is called a sub-problem. Hopefully, the optimal solution of the i^{th} sub-problem $g^i(\pi)$ lies in the PS (PF) of the original problem.

Neighborhood: The neighborhood $B_i = (i_1, i_2, \ldots, i_K)$ of the i^{th} sub-problem contains the indices of similar sub-problems, i.e., the i_j^{th} ($j = 1, \ldots, K$) sub-problems are the most similar ones to the i^{th} sub-problem.

Sub-problem definition: In [16] Tchebycheff approach is used to define the sub-problems as given follows:

$$\min g^i(\pi) = g\left(\pi|\lambda^i, z^*\right) = \max_{1<j\leq m} \lambda_j^i \left| f_i(\pi) - z_j^* \right|$$

Where $\lambda_i = (\lambda^i_1, \ldots, \lambda^i_m)^T$ is a weight vector with the i^{th} sub-problem, $z* = (z*1, \ldots, z*m)^T$ is a reference point. Reference point weakly dominates all the other solutions in the population. It is clear that all the sub-problems are with the same form and can only be differentiated by the weight vectors. If two vectors are close to each other, the corresponding sub-problems should be similar to each other and their optima should also be close in both the decision and objective spaces in most cases. By using the weight vectors, the neighborhood could be determined before the algorithm execution. A Probabilistic model Pi stores information extracted from the population for the i^{th} sub-problem and mating process will be continued. MOEA/D framework follows.

MOEA/D Algorithm framework

The following steps are followed in the MOEA/D algorithm framework [18] [16]. First MOEA/D converts an MOTSP into N sub-problems and randomly generates a solution for each sub-problem. Then initialize the reference point $z*$ followed by Initialize the weight vectors, neighborhood and probability matrix for each sub-problem. For each sub-problem i = 1,..., N, do Sample a new solution be a unvisited city by randomly selecting according to the probability and repeat the process until the whole tour is constructed. Then do update of reference point, Update solutions Set and Update probability model. If the stopping conditions are satisfied, then stop; otherwise do the process again. Algorithm 1 explains the MOEA/D working process.

IIOAB JOURNAL
ISSN: 0976-3104

Algorithm 1: MOEA/D

Input: N: number of SOSPs (scalar optimization sub-problems) ; W: number of the neighbors for each SOSP; $\lambda^1, \ldots, \lambda^N$: uniformly distributed weight vectors; pc: crossover rate; pm: mutation rate.

Initialization
1. Set EP = Ø (External Population)
2. For each _i, calculate the W closest weight vectors, $\lambda^{i(1)}, \ldots, \lambda^{i(W)}$, by Euclidean distance and set φ(i) = {i(1), . . ., i(W)}.
3. Generate an initial population $x_1, \ldots x_N$ and evaluate fu(xj)for each individual
4. Initialize z = (z1,..., zm)
Repeat
5. For i = 1 to N do
Reproduction:
6. Generate a new solution y by two individuals x_u and x_l using crossover and mutation operators, where u, l ∈ φ(i)
Improvement:
7. Improve y by using a problem-specific improvement repair operator, which is optional.
Update of z:
8. For j = 1,..., m, if fj(y) < zj, set zj= fj(y)
9. Update of neighboring solutions
Update of EP:
10. Remove those solutions dominated by y from EP and add y to EP if it is not dominated by any member in EP
Termination:
11. Until stopping criteria are satisfied, output EP

In [19], Algorithm decomposes the population into 's' scalar optimization sub-problems according to the Tchebycheff approach. It follows a two-chromosome representation for individual's representation. The first chromosome locates the cities while the second chromosome indicates which vehicle is to be assigned to visit the city specified in the first chromosome, which improves the performance of the algorithm. A chemical reaction optimization based decomposition method is introduced in [6]. It follows the same steps as MOEA/D algorithm framework; in addition to that a chemical collision stage is followed after initialization stage. A Parallel Procedure for Dynamic Multi-objective TSP [21] follows the decomposition method to decompose the problems and follow the parallel execution of the sub-problems for time efficient process. In [21] objectives are represented in the form of matrix, in which we can increase the number of objectives for optimization. But the number of objectives affects directly the execution time. In [31] different approaches in MOEA/D are listed to solve the MOTSP.

Multi-objective Genetic Algorithm

Multi Objective Genetic Algorithm (MOGA) was given by Fonseca et al [4] [13]. It is an extension of single objective optimization algorithm. The rank to an individual is assigned based on the number of solutions in the population by which it is dominated. GA has two main parts, an evolution function and a fitness function. In the case of the MOTSP, the parameters produced by the evolution function might be the order of the nodes through which the path will go. The fitness function in that same case would return the total length of the path found. The GA would then compare fitness values for each input string and assign priority to the ones that returns lower path lengths and other objectives. Based on the objective values the best solutions will be produced. The framework of the MOEA was explained in [14] [24]. By following it MOGA steps are explained below.

Framework of MOGA

MOGA starts with initial population and assign the fitness value to the individuals and apply the Evolution operator on the individuals. Do this process until satisfaction criteria met and finally it will produce the best path for MOTSP. Algorithm 2 explains the MOGA working process.

Algorithm 2: MOGA

Input: $Population_{size}$, $Problem_{size}$, $P_{crossover}$, $P_{mutation}$
Output: S_{best}
Population <- InitializePopulation ($Population_{size}$, $Problem_{size}$)
EvaluatePopulation (Population)
S_{best} <- GetBestSolution (Population)
While (~StopCondition())
    Parents <- SelectParents (Population, $Population_{size}$)

IIOAB JOURNAL
ISSN: 0976-3104

```
        Children <- Ø
        For (Parent1, Parent2 € Parents)
                Child1, Child2 <- Crossover (Parent1, Parent2, Pcrossover)
                Children <- Mutate (Child1, Pmutation)
Children <- Mutate (Child2, Pmutation)
        End
        EvaluatePopulation (Children)
        Sbest <- GetBestSolution (Children)
        Population <- Replace (Population, Children)
End
Return (Sbest)
```

GA can solve problems with non-parametrical problems, multi-dimensional, non-continuous and non-differential optimization problems. But Genetic Algorithm has weaker local search ability [14]. During the later period of Hierarchical Genetic Algorithm, the fitness converges, and less superior individuals are produced. However, Hybrid Simulated Annealing Algorithm can make it jump out of the erroneous zone of local optimum. Due to the compatibility of Genetic Algorithm, it is feasible to combine Simulated Annealing Algorithm and Hierarchical Genetic Algorithm to form the modified Simulated Annealing Genetic Algorithm [8]. The modified Simulated Annealing Genetic Algorithm can sooner achieve a better global optimum solution. The reasons are: the suffix structure design of chromosome reduced the space of the solution, the self-adaptive genetic operator and double crossover and mutation improved 'premature convergence problem'; the introduction of Simulated Annealing Algorithm stretched the fitness and enhanced the local search ability.

Multiple traveling salesman problem was solved using GA in [5]. Two types of GAs were developed in it. The first one is a multi-objective GA that uses the sum of the route distance of each salesman to estimate the overall distance, and the standard deviation of the routes to estimate the balance. The second is a mono-objective GA with a fitness function that combines these two objectives by the use of a parameter. GA is modified into discrete GA and combined with fuzzy technique [34] in order to solve multi-objective problem. The proposed algorithm is able to find better result in shorter computational time. Xiaomei Sun [69] solved the Intelligent Transportation Systems (ITS) using improved Genetic Algorithm. It divides the population into subgroups and do the selection operation and recombine the subgroups followed by crossover and mutation operation. In [19] MOGA is used to solve vehicle routing problem.

Non-dominated Sorting Genetic Algorithm-II (NSGAII)

Non-dominated Sorting Genetic Algorithm-II (NSGAII) was introduced by Deb et al. [13]. It is an improved version of NSGA. The rank of every solution is computed based on how many number of solutions it dominates. In order to maintain the diversity of a population the algorithm finds average distance of two neighbors on either side of a solution along each of the objectives. The calculated distance is called crowding distance of that solution. For generating mating pool for next generation, selection of solutions is performed based on rank and crowding distance. The concept of dominance is used to find the best individual. It follows:

Concept of dominance

The concept of dominance is applied to multi-objective problems to compare two solution candidates $X1$, $X2$, and determine if a solution dominates the other one. In particular, the dominance is a method for the classification of the solutions which ensures the selection of the best solution in the resulting population Rt.

Definition [6]: Given two solutions $X1$ and $X2$, solution $X1$ dominates solution $X2$, if the following conditions are satisfied:
1. Solution $X1$ is not worse than $X2$ for all the objectives;
2. Solution $X1$ is strictly better than $X2$ for at least one objective.

Using these dominance concept individuals are ranked. When two solutions have the same rank then a solution that has higher crowding distance is selected for mating. The algorithm selects the solutions for the next generation based on following policy: select best solutions out of union of the best parents and best offspring (obtained after application of genetic operators). The following criteria are used for selection of the best solutions from the union: fitness and spread. As the algorithm selects the best solutions from the union, it does not require extra memory to preserve elite solutions. The NSGA II algorithm framework follows [27] [28] [29].

NSGA II Algorithm Framework

It starts with the initial population and assigns the rank for each individual. Then crowding distance is calculated for each individual. Based on the rank and crowding distance of the individuals best solutions are selected for crossover and mutation operations. Until the stopping criteria met the steps are repeated.

IIOAB JOURNAL
ISSN: 0976-3104

Algorithm 3: NSGA II

Input: Population $_{size}$, problemSize, P $_{crossover}$, P $_{mutation}$
Output: children

```
Population <- InitializePopulation (Population_Size , ProblemSize)
EvaluateAgainstObjectiveFunctions(Population)
FastNondominatedSort(Poplation)
Selected <- SelectParentsByRank(Population, Population size )
Children <- CrossoveAndMutation (Selected, P crossover, P mutation)
While (~ StopCondition())
        EvaluateAgainstObjectiveFunctions(Children)
        Union <- Merge (Population, Children)
        Fronts <- FastNondominatedSort (Union)
        Parents <- Ø
Front L<- Ø

For (Front i € Fronts)
CrowdingDistanceAssignment (Front i)
If (Size (Parents) + Size (Fronts i) > Population size)
Front_L <- i
Break()
      Else
Parents <- Merge (Parents, Front i)
      End
End
If (Size (Parents) + Size (Fronts i) < Population size)
Front L <- SortByRankAndDistance (Front L)
        For (P1 to P_Population size – SizeFront L)
    Parents <- Pi
End
End
Selected <- SelectPrentByRankAndDistance (Parents, Population Size)
Population <- Children
Children <- CrossoverAndMutation(Selected, P_crossover, P_mutation
End
Return (Children)
```

In [6], Author proposed a methodology based on the non-dominated sorting genetic algorithm NSGA-II to solve the Multiple TSP. It follows the same algorithm framework given above but the number of salesman will be more than one. An improved NSGA II is implemented in [40]. Specifically, a layer strategy according to need is proposed to avoid generating unnecessary non-dominated fronts. The arena's principle is also adopted to construct non-dominated set, so as to reduce the count of dominance comparison. In addition, an order crossover like operator and an inversion mutation operator are adopted for MOTSP.

Non-dominated sorting differential evolution algorithm for the minimization of route based fuel consumption multi-objective vehicle routing problems is introduced in [31] [29]. It uses the hybrid version of NSGA II to solve the routing problem by optimizing the fuel consumption. MOTSP is solved using two algorithms based on Differential Evolution, and the third one is based on NSGA II. The algorithms solve 2 to 5 objective functions TSP in [27] [28]. Vehicle routing problem with uncertain travel cost is solved using NSGA II in [5] [48].

Multi-objective Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is a population-based meta-heuristic inspired by the pheromone trail laying and following behavior of some real ant species. ACO was originally designed for solving single-objective combinatorial optimization, and later it has proven to be one of the most successful algorithms for modeling discrete optimization problems. Due to notable results on these applications, ACO algorithms were soon extended to tackle problems with more than one objective (MOPs), called multi-objective ACO (MOACO), most of these algorithms have different design choices and focus in terms of Pareto optimally, that is, they do not make a priori assumptions about the decisions maker's preferences. So it gives the best result in real time problem solving.

ACO implementation mainly consists of two stages: tour construction and pheromone update. The process of tour construction and pheromone update is applied iteratively until a termination condition is met (such as a set number of iterations). Both the tour construction and pheromone update stages can be performed independently [17]. The main idea in this algorithm is that the behavior of each individual ant produces an emergent behavior in the colony. When applied to the MOTSP, individual agents ("ants") traverse the graph of the problem, leaving a chemical (pheromone) trail behind them. At each node it comes to, an ant

IIOAB JOURNAL
ISSN: 0976-3104

must decide which edge to take to the next node. This is done by checking each edge for pheromone concentration and applying a probability function to the decision of which edge to choose. The higher the concentration of pheromone, the more likely the ant is to choose that edge. Also, to avoid stagnation in travel, the pheromone is given an evaporation rate so that at each iteration the pheromone loses a certain percentage on each edge. The MOACO algorithm framework is explains the above given steps to solve MOTSP.

MOACO Algorithm Framework

The algorithm maintains: (I) $\tau^1, \ldots, \tau^K$, where τ^j is the current pheromone matrix for group j, storing its learned knowledge about the sub-region of PF that it aims at approximating; (II) $\eta 1, \ldots, \eta^N$, where η^i is the heuristic information matrix for sub-problem i, which is predetermined before the construction solution starts; (III) EP, which is the external archive containing all the non-dominated solutions found so far. Algorithm 4 explains the working process of MOACO.

Algorithm 4: MOACO

```
Define Original pheromone
        For iteration=1 to i= (1,2,3...n)
          For ant=1 to i = (1,2,3...n)
            Random start node
                Do while node < i
                    Select next node
                Loop
            Next End Loop ant
            Calculate Multi – objective function
    For ant=1 to i=(1,2,3...n)
                Do while node < i
                   Update pheromone
                Loop
                Checking pheromone upper and pheromone lower
          Next End Loop ant
        Next End loop iteration
```

All the problems using MOACO follow the above MOACO algorithm framework. MOACO can be combined with Genetic Algorithm and MOEA/D to give better performance. In [42], a hybridized algorithmic approach to solve 4- dimensional Travelling Salesman Problem is introduced. The algorithm is a hybridization of rough set based ant colony optimization (rACO) with genetic algorithm (GA). In this the initial solutions are produced by ACO which act as a selection operation of GA and then GA is developed with a virgin extended rough set based selection (7-point scale), comparison crossover and generation dependent mutation. The development of rACO-GA is in general form and it can be applied in other discrete problems such as network optimization, graph theory, solid transportation problems, vehicle routing, covering salesman problem, VLSI chip design, industrial information integration, information management, supply chain, airline industry, etc. ACO is combined with PSO and used to solve TSP which gives good result [2].

In [30] [24] ACO is implemented to solve multi objective problems combined with Genetic Algorithm. Following other MOEA/D-like algorithms, MOEA/D-ACO decomposes a multi-objective optimization problem into a number of single-objective optimization problems. Each ant (i.e., agent) is responsible for solving one sub-problem. All the ants are divided into a few groups, and each ant has several neighboring ants. An ant group maintains a pheromone matrix, and an individual ant has a heuristic information matrix. During the search, each ant also records the best solution found so far for its sub-problem. To construct a new solution, an ant combines information from its group's pheromone matrix, its own heuristic information matrix, and its current solution. An ant checks the new solutions constructed by itself and its neighbors, and updates its current solution if it has found a better one in terms of its own objective. ACO is used to solve Multi-objective Bus Route Planning.

Multi-objective Particle Swarm Optimization (MOPSO)

Particle Swarm Optimization (PSO) was given by (Kennedy and Eberhart 1995) [15]. This method is inspired by the social behavior of animals living swarm. Initially, they tried to simulate the ability of birds to fly synchronously and their ability to change direction suddenly while remaining optimal training. The particles are individuals and they move into hyperspace research based on limited information:

1. Each particle has a memory which enables to store the best point at which it has already passed and it tends to return to that point.
2. Each particle is informed of the best known point in its neighborhood and it will tend to go to that point.

The movement of a particle tends to follow her go her own way or particle tends to return to the best site in which it has already passed or the Particle tends to move toward the best ever achieved by its neighbors. The position of each particle is modified according to its own experience and that of its neighbors. To make the PSO capable to handle MOP the following modifications are done.

IIOAB JOURNAL
ISSN: 0976-3104

General modifications on PSO to handle MOPs are: (1) External Archive Maintenance (2) Select Global Leaders (3) Update Personal Best (4) Mutation Operator (Perturbation). PSO produce a single solution and the solution of a problem with multiple objectives is not a single solution (as in global optimization). Instead, in the multi-objective optimization, we aim to find different solutions (called Pareto Optimal Set) using MOPSO. By following these steps MOPSO framework is explained below.

MOPSO framework

Initialize swarm population and velocity. Do the Fitness evaluation and Pareto dominance for ranking particles (solutions). Based on the ranking of individuals Personal Best (Pbest) swarm population is stored on to the memory and non-dominated solutions are stored in External Archive. Particles Select global leaders from External Archive and compute PSO equation. Using fitness rank individuals are mutated. The new Personal Best are updated and maintained on external archive. If satisfaction criteria met then stop the process else continue [15]. Algorithm 5 explains the working of MOPSO.

Algorithm 5: MOPSO

```
Begin
Initialize Swarm
Initialize leaders in an external Archive
Quality (Leaders)
g=0;
While g < gmax
    For each Particle
            Select leader
            Update Position
            Mutation
Evaluation
Update pbest
   End For
Update leaders in the external Archive
Quality (leaders)
g++
End while
Report results in the external Archive
End
```

In [15] MOTSP is solved using MOPSO. In multi-objective Particle Swarm Optimization (MOPSO), a ranking operator is applied to the population in a predefined iteration to build an initial archive using €-dominance to select the best solutions for mating. It uses the advantage of Pareto approach which is based on the concept of external archive with the rapidity of convergence of PSO as to minimize the total distance traveled by a particle and minimize the total time.

RESULTS

This section studies the effectiveness of the listed five MOEA algorithms. Then, we evaluate the overall performance of MOEA algorithms by comparing its Performance measures. The associated parameters are defined below.

Performance measures

To evaluate the performance of the MOEA algorithms, we employ the following widely recognized performance measuring metrics. Let PF_{ref} be a reference set of solutions well approximating the true PF, and PF_{known} be the set of non-dominated solutions obtained by an algorithm.

Approximation set [34]: An approximation set is defined by Zitzler et al. as follows: let A $\in H$ be a set of objective vectors. A is called an approximation set if any element of A does not dominate or is not equal to any other objective vector in A. The set of all approximation sets is denoted as Z. The result of solving a real-world problem usually is an approximation set A and not the Pareto optimal front PF*.
Cardinality metrics: the cardinality of A refers to the number of solutions that exists in A. Intuitively; a larger number of solutions are preferred.
Accuracy metrics: this aspect refers directly to the convergence of A. In other words, it indicates how distant A is from the theoretical Pareto optimal front PF*. Notice that when the Pareto optimal front is unknown, a reference set R is considered instead.
Diversity metrics: distribution and spread are two very closely related facets. The distribution refers to the relative distance among solutions in A, while the spread refers to the range of values covered by the solutions in A. The spread is also known as "the extent" of an approximation set.

IIOAB JOURNAL
ISSN: 0976-3104

Maximum spread (MS) [34]: MS reflects how well the true PF is covered by the points in PF_{known} through the hyperboxes formed by the extreme function values observed in PF_{ref} and PF_{known}.
Hypervolume (HV): HV also known as S metric, hyper-area is an unary metric that measures the size of the objective space covered by an approximation set. A reference point must be used to calculate the mentioned covered space. HV considers all three aspects: accuracy, diversity and cardinality.
Inverted generational distance (IGD) [34]: IGD is defined as the average distance from each point v in PF_{ref} to its nearest counter-part in PF_{known}, as follows:

$$IGD = \frac{\sum v \epsilon PF_{ref} d(v, PF_{known})}{|PF_{ref}|}$$

Where d(v, PF_{known}) is the Euclidean distance (in the objective space)between solution v in PF_{ref} and its nearest solution in PF_{known} and $|PF_{ref}|$ is the number of solutions in PF_{ref}. IGD measures the convergence and diversity of an obtained non-dominated solution set. This metric is commonly used and a lower IGD indicates a better overall performance of an algorithm.
Average Computational Time (ACT): ACT is the average running time consumed by an algorithm over 30 runs. This metric is a direct indicator of the computational complexity of an algorithm.

Result comparison

IGD reflects the overall performance of an algorithm regarding the quality of the obtained PF_{known}. So we compared all the listed MOEA algorithms using IGD. We computed the IGD of the algorithms for 10 runs with randomly generated cities. The IGD of MOEA/D, NSGA II, MOGA, MOACO and MOPSO algorithms are shown in the [Table 1]. RGC represents Randomly Generated Cities.

Table 1: Results of IGD of the MOEA algorithms

RGC/ Algorithms	MOEA/D	NSGA II	MOGA	MOACO	MOPSO
1	0.00	0.00	0.89	1.50	2.07
2	0.00	0.29	1.33	1.72	2.39
3	0.00	0.50	1.42	2.06	2.46
4	0.54	1.08	1.68	2.38	2.48
5	0.77	1.22	2.01	2.26	2.56
6	0.80	1.50	2.10	2.53	2.63
7	0.76	1.36	2.34	2.56	2.96
8	0.62	1.25	2.56	3.06	3.76
9	0.17	1.46	2.60	3.16	3.96
10	0.20	1.37	2.50	3.28	4.45

As mentioned in the IGD definition, lower IGD indicates a better overall performance of an algorithm. From the [Table 1], it is clear that MOEA/D is having the lower IGD in more number of runs compared to other algorithms, then followed by NSGA II, MOGA, MOACO and MOPSO algorithms. The below [Fig. 2] shows the result of IGD metric performance of the five algorithms. From the [Fig. 2] it is observed that when the number of cities is increasing, the IGD of NSGA, MOGA, MOACO and MOPAS algorithms are also increasing and the starting value of IGD is also very high. But in MOEA/D the IGD value is zero and when the number of cites is increasing value of IGD is going in a constant way. So from the definition of IGD, it is known that MOEA/D is having good diversity and convergence. All the other four algorithms are lacking in their performance with respect to IGD. So with respect to IGD, MOEA/D is giving best performance.

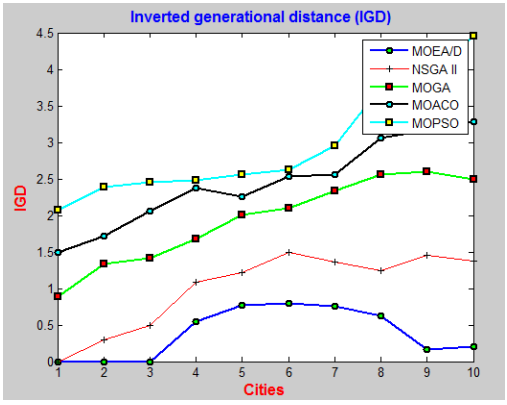

Fig. 2: IGD of MOEA algorithms.

The Average Computational Time (ACT) is also calculated to find the best algorithm among the five to solve MOTSP with respect to time complexity. The ACT of the MOEA are listed in the below [Table 2]. ACT is measure in seconds.

IIOAB JOURNAL
ISSN: 0976-3104

Table 2: Average Computational time (ACT) *(Sec)* of MOEA Algorithms

RGC/Algorithm variants	MOEA/D	NSGA II	MOGA	MOACO	MOPSO
1	5.63	15.31	18.80	23.75	26.83
2	7.05	23.53	25.45	35.68	26.04
3	17.09	61.52	50.28	56.57	27.41
4	34.23	60.73	102.28	134.85	179.02
5	34.24	93.42	105.78	170.47	228.61
6	38.83	120.38	170.03	230.78	240.99
7	41.44	196.29	240.09	399.76	503.51
8	89.21	266.27	450.34	630.54	782.52
9	134.92	747.23	876.22	934.23	1027.34
10	484.71	958.91	1021.34	1194.25	1324.86

For the [Table 2], MOEA/D is having the minimum computational time. NSGA II algorithm took almost three times more ACT than MOEA/D. Other algorithms are giving even higher time complexity. The ACT result of the algorithms is shown in the [Fig. 3]. When the number of cities is increasing the ACT also increases. MOEA/D has the lower computational time compared to other algorithms. So with respect to ACT also, MOEA/D is the best algorithm to solve MOTSP.

From the result analysis, MOEA/D is the best algorithm to solve MOTSP in all aspects. NSGA II is also giving good performance only but it is good when the number of objectives considered is only two [11] [13]. MOGA is giving good performance for stable environment but not for dynamic environment and the local search ability is also not good [35] [13].

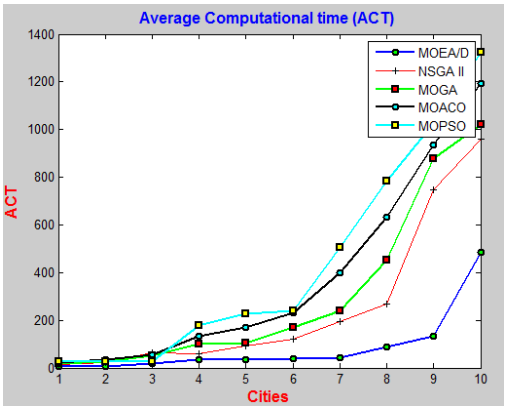

Fig. 3: ACT (sec) of MOEA Algorithms.

MOACO is performing well in dynamic environment but in ACO starting and destination nodes should be defined at earlier. MOPSO is good in global search but the convergence and local search are not at the expected level to solve MOTSP. MOEA/D is giving better performance in all aspect compared to other algorithms which is proved from the IGD and ACT analyses of the algorithms [18] [11] [17] [16].

CONCLUSION

In this paper the problem of TSP is explained in order to understand the concept and need of MOTSP which optimizes more objectives to find the best solutions. There are many methods to solve the MOTSP. Among those Multi-Objective Evolutionary Algorithms are the best methods to solve it efficiently. By evolving a population of solutions, multi-objective evolutionary algorithms (MOEAs) are able to approximate the Pareto optimal set in a single run. It results in nearest optimal solution within a reasonable time by optimizing many objectives simultaneously. The MOEA algorithms used to solve MOTSP are explained with its algorithm framework in section II. Each and every algorithm is best suited in some situation or environment to solve it. Among listed MOEA algorithms MOEA/D is giving the best solution for the MOTSP.

CONFLICT OF INTEREST

There is no conflict of interest.

ACKNOWLEDGEMENTS
This work is a part of the Research Projects sponsored by Visvesvaraya Ph.D. Scheme for Electronics & IT, Ministry of Electronics & Information Technology, India, and Reference No: PHD-MLA-4(44)/2015-16, dated August 2015. The authors would like to express their thanks for the financial supports offered by the Sponsored Agency.

FINANCIAL DISCLOSURE
None

REFERENCES

[1] Saraei M, Analouel R, Mansourl P. [2015] Solving of Travelling Salesman Problem using Firefly Algorithm with Greedy Approach, (June).
[2] Khanra A, Maiti MK, Maiti M. [2015] Profit maximization of TSP through a hybrid algorithm. Computers and Industrial Engineering, 88:229–236. https://doi.org/10.1016/j.cie.2015.06.018
[3] Fdhila R. [2014] Distributed MOPSO with dynamic Pareto Front driven population analysis for TSP problem, 294–299.
[4] Changdar C, Mahapatra GS, Kumar Pal. [2014] An efficient genetic algorithm for multi-objective solid travelling salesman problem under fuzziness. Swarm and Evolutionary Computation, 15:27–37. https://doi.org/10.1016/j.swevo.2013.11.001
[5] Lopes CR. [2015] Using Genetic Algorithms to minimize the distance and balance the routes for the multiple Traveling Salesman Problem, 3171–3178.
[6] Ruben Ivan Bolapos, Mauricio Granada Echeverry, John Willmer Escobar. [2015] A multiobjective non-dominated sorting genetic algorithm [NSGA-II] for the Multiple Traveling Salesman Problem, Decision Science Letters, 4: 559–568. https://doi.org/10.5267/j.dsl.2015.5.003
[7] Segura C, Coello CA, Miranda G, Leon C. [2017] Using multi-objective evolutionary algorithms for single-objective constrained and unconstrained optimization. Annals of Operations Research, 240(1): 217–250. https://doi.org/10.1007/s10479-015-2017-z
[8] Xu M, Li S, Guo J. [2017] Optimization of Multiple Traveling Salesman Problem Based on Simulated Annealing Genetic Algorithm, 25.
[9] Sidoti D, Avvari GV, Mishra M, Zhang L, Nadella BK, Peak JE, Pattipati KR. [2016] A Multiobjective Path-Planning Algorithm With Time Windows for Asset Routing in a Dynamic Weather-Impacted Environment. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 1–16. https://doi.org/10.1109/TSMC.2016.2573271
[10] Bazgan C, Gourvès L, Monnot J, Pascual F. [2013] Single approximation for the biobjective Max TSP. Theoretical Computer Science, 478, 41–50. https://doi.org/10.1016/j.tcs.2013.01.021
[11] Köksalan M, Tezcane Öztürk D. [2017] An evolutionary approach to generalized biobjective traveling salesperson problem. Computers and Operations Research, 79: 304–313. https://doi.org/10.1016/j.cor.2016.04.027
[12] Li M, Yang S, Liu X. [2015] Bi-goal evolution for many-objective optimization problems. Artificial Intelligence, 228: 45–65. https://doi.org/10.1016/j.artint.2015.06.007
[13] Rahimi M, Baboli A. [2014] A bi-objective inventory routing problem by considering customer satisfaction level in context of perishable product.
[14] Vachhani VL, Prajapati HB. [2015] Survey of Multi Objective Evolutionary Algorithms.
[15] Jiang J, Gee SB, Arokiasami, WA, Tan KC. [2014] Solving Vehicle Routing Problem with Stochastic Demand Using Multi-objective Evolutionary Algorithm, 1. https://doi.org/10.1109/ISCMI.2014.18
[16] Zhou A, Gao F, Zhang G. [2013] A decomposition based estimation of distribution algorithm for multiobjective traveling salesman problems. Computers and Mathematics with Applications, 66(10):1857–1868. https://doi.org/10.1016/j.camwa.2013.05.031
[17] Ke L, Zhang Q, Battiti R. [2013] MOEA/D-ACO: A multobjective evolutionary algorithm using decomposition and AntColony. IEEE Transactions on Cybernetics, 43(6): 1845–1859. https://doi.org/10.1109/TSMCB.2012.2231860
[18] Ke L, Zhang Q, Battiti R. [2014] Hybridization of decomposition and local search for multiobjective optimization. IEEE Transactions on Cybernetics, 44(10): 1808–1820. https://doi.org/10.1109/TCYB.2013.2295886
[19] Souza MZ De, Trinidad A Pozo R. [2014] A GPU Implementation of MOEA / D-ACO for the Multiobjective Traveling Salesman Problem, 6–11. https://doi.org/10.1109/BRACIS.2014.65
[20] GAO F, Zhou A, Zhang G. [2012] An Estimation of Distribution Algorithm based on Decomposition for the Multiobjective TSP, (Icnc), 817–821.
[21] Shim VA. (2012). A Hybrid Estimation of Distribution Algorithm for Solving the Multi-objective Multiple Traveling Salesman Problem, 10–15.
[22] Li W. [2012] A Parallel Procedure for Dynamic Multi-objective TSP. https://doi.org/10.1109/ISPA.2012.10
[23] Xing H, Wang Z, Li T, Li H, Qu R. [2017] An improved MOEA/D algorithm for multi-objective multicast routing with network coding. Applied Soft Computing Journal, 59:88–103. https://doi.org/10.1016/j.asoc.2017.05.033
[24] Kuo RJ. [2017] A Fuzzy Multi-Objective Vehicle Routing Problem for Perishable Products Using Gradient Evolution Algorithm.
[25] Psychas ID, Delimpasi E, Marinakis Y. [2015] Hybrid evolutionary algorithms for the Multiobjective Traveling Salesman Problem. Expert Systems with Applications, 42(22):8956–8970. https://doi.org/10.1016/j.eswa.2015.07.051
[26] Psychas ID, Delimpasi E, Marinakis Y. [2015] Hybrid evolutionary algorithms for the Multiobjective Traveling Salesman Problem. Expert Systems with Applications, 42(22):8956–8970. https://doi.org/10.1016/j.eswa.2015.07.051
[27] Psychas ID, Marinaki M, Marinakis Y, Migdalas A. [2016] Non-dominated sorting differential evolution algorithm for the minimization of route based fuel consumption multiobjective vehicle routing problems. Energy Systems. https://doi.org/10.1007/s12667-016-0209-5
[28] Luo Y, Liu M, Hao Z, Liu D. [2014] An Improved NSGA-II Algorithm for Multi-objective Traveling Salesman Problem, 12(6):4413–4418. https://doi.org/10.11591/telkomnika.v12i6.5476
[29] GarciaNájera A, Bullinaria JA, Gutiérrez-Andrade MA. [2015] An evolutionary approach for multi-objective vehicle routing problems with backhauls. Computers & Industrial Engineering, 81, 90–108. https://doi.org/10.1016/j.cie.2014.12.029
[30] Bederina H. [n.d.] Evolutionary Multi-Objective Optimization Approach for the Vehicle Routing Problem with Uncertain Travel Time.
[31] Mathew N, Smith SL, Waslander SL. [2015] Planning Paths for Package Delivery in Heterogeneous Multirobot Teams. IEEE Transactions on Automation Science and Engineering, 12(4): 1298–1308. https://doi.org/10.1109/TASE.2015.2461213
[32] Maity S, Roy A, Maiti M. [2017] Journal of Industrial Information Integration An intelligent hybrid algorithm for 4- dimensional TSP. Journal of Industrial Information Integration, 5:39–50. https://doi.org/10.1016/j.jii.2017.02.001
[33] Pierre DM, Zakaria N. [2017] Stochastic partially optimized cyclic shift crossover for multi-objective genetic algorithms for the vehicle routing problem with time-windows. Applied Soft Computing Journal, 52:863–876. https://doi.org/10.1016/j.asoc.2016.09.039
[34] Riquelme N, Von LC. [2015] Performance metrics in multi-objective optimization, 1.