Journal of Basic and Applied Engineering Research
p-ISSN: 2350-0077; e-ISSN: 2350-0255; Volume 2, Number 18; July-September, 2015, pp. 1612-1616

http://www.krishisanskriti.org/Publication.html

Review of Algorithms to Solve Travelling Salesman Problem

Avani Sharma[1] and Kamlesh Dutta[2]

[1,2]Department of Computer Science and Engineering National Institute of Technology, Hamirpur Himachal Pradesh-177005, India
E-mail: [1]avaninith@gmail.com, [2]kdnith@gmail.com

Abstract—*The travelling salesman problem (TSP) is a well-known problem in which shortest route of the salesman covering all cities once and returning to the starting point, is to be determined. This is done either by using the exact algorithms or heuristic algorithms. The main concern with the exact algorithms is their complex nature to solve the combinatorial travelling salesman optimization problem. This problem is tackled by the various heuristic algorithms which aim at finding the optimal route for the salesman. This paper presents a review of different algorithms to solve TSP and find the shortest route through all the cities that the salesman has to travel.*

1. INTRODUCTION

The travelling salesman problem (TSP) is one of the most intensively studied problems in the combinatorial optimization problems [1]. An optimal path has to be determined in terms of time and monetary value through a number of cities which is to be travelled by the salesman to reach his destination in such a manner that he travels each city exactly once and return to the origin city. The TSP is a NP (non-deterministic polynomial time) hard problem because any algorithm for solving it can be translated into one to solve NP problem.

Many researchers have tried to solve TSP using the exact algorithms which includes branch & cut algorithms[2], Dynamic Programming [3] and evolutionary algorithms such as genetic algorithm (GA) [4], particle swarm optimization (PSO) [5], artificial bee colony (ABC) algorithm [6], ant colony optimization (ACO) [7, 8] etc. The exact algorithms find the exact short route in exponential number of steps but pose the difficulty of complexity and high demand of computer power. The researchers use the evolutionary algorithms to construct and improve the route to travel.

This paper gives the review of different algorithms used to solve the TSP and a detailed description of the most recently introduced hybridized Honey Bees Mating Optimization (HHBMO) algorithm [9]. The hybrid HBMO incorporates a number of different procedures in each step of the main algorithm in order to increase the efficiency of the proposed algorithm [10]. The author in [10] presents the remarkable results using the proposed algorithm to solve the TSP. The hybrid HBMO takes less computational time making the algorithm faster and more efficient for solving large-scale problems.

2. CLASSICAL ALGORITHMS TO SOLVE TRAVELLING SALESMAN PROBLEM (TSP)

The different exact algorithms such as Branch and Bound and Dynamic Programming are used to solve Travelling Salesman Problem.

2.1 Branch and Bound

General technique for branch and bound algorithms involves modeling the solution space as a tree and then traversing the tree exploring the most promising sub trees first [2]. This is continued until either there are no sub trees into which to further break the problem, or we have arrived at a point where, if we continue, only inferior solutions will be found.

Algorithm:

search(1,r,best)

pre: t=solution space tree

r=vertex in t

best=best solution found so far

post: best=best solution found after searching sub tree rooted at r

if r is a complete solution more optimum than best then best=r

generate the children of r

computer bounds for vertices in sub trees of children

V_1, ... ,V_2 *= feasible children with good lower bounds*

for i : = 1 to k

if V_i *has a promising upper bound*

then search(t, V_i*, best)*

Branch-and-Bound algorithm, can be used to process TSPs containing 40–60 cities.

2.2 Dynamic Programming (DP)

Dynamic programming is a very powerful technique for efficiently computing recurrences by storing partial results and reusing them when needed [3]. It is a method for solving a complex problem by breaking it down into a collection of simpler sub problems. It demands very elegant formulation of the approach and simple thinking and the coding part is very easy. The idea is very simple, If you have solved a problem with the given input, then save the result for future reference, so as to avoid solving the same problem again, shortly 'Remember your Past'. If the given problem can be broken up in to smaller sub-problems and these smaller sub problems are in turn divided in still-smaller ones, and in this process, if you observe some over-lapping sub problems, then it is a big hint for DP. Also, the optimal solutions to the sub problems contribute to the optimal solution of the given problem (referred to as the Optimal Substructure Property).

There are two ways of doing this.

1. *Top-Down:* Start solving the given problem by breaking it down. If you see that the problem has been solved already, and then just returns the saved answer. If it has not been solved, solve it and save the answer. This is usually easy to think of and very intuitive. This is referred to as *Memoization*.
2. *Bottom-Up:* Analyze the problem and see the order, in which the sub-problems are solved and start solving from the trivial sub problem, up towards the given problem. In this process, it is guaranteed that the sub problems are solved before solving the problem. This is referred to as *Dynamic Programming*.

Steps followed while implementing Dynamic Programming:

1. Characterize the recursive structure of an optimal solution,
2. Define recursively the value of an optimal solution,
3. Compute, bottom up, the cost of a solution,
4. Construct an optimal solution.

This approach is also used to solve travelling salesman problem but only for limited number of cities.

3. HEURISTIC ALGORITHMS TO SOLVE TRAVELLING SALESMAN PROBLEM (TSP)

The different optimization algorithms such as genetic algorithm (GA), particle swarm optimization (PSO) , artificial bee colony (ABC) algorithm, ant colony optimization (ABC) used to solve the TSP are described in this section.

3.1 Genetic algorithm (GA)

The process of finding optimal solution using genetic algorithm consists of representing the cities (individuals) in the form of chromosomes which is in the form of matrices in case of TSP [4]. The individual cities with the best genetic material (minimum time and monetary value) are found out using the operators: crossover and mutation. The crossover operator helps to increase the average quality of the population and mutation operator helps in avoiding local optima and producing new paths. A fitness function is used to evaluate the quality or probability of the individual path in the population. The problem with the GA is that the representation of the path is not unique. Sometimes, the crossover operator also does not produce valid tour paths. Algorithm steps are as follows:

1. Initialize the population of chromosomes which are suitable solutions for the problem.
2. Evaluate the fitness of the defined objective function of each chromosome in the population.
3. A new population is generated by using the following operators.

   a) *Selection*: Select two parent chromosomes from a population according to their fitness evaluated in Step 2. The probability of selection of a parent chromosome is higher with the better fitness value.

   b) *Crossover*: The new offspring are generated using the crossover operator. The crossover probability controls the number of chromosomes undergoing crossover to produce new offspring.

   c) *Mutation*: The population is allowed to mutate with a low mutation probability. The mutation probability controls the probability with which new genes are introduced into the population for trial.

   d) Replace this mutated population in place of the initial generated populated in Step 1.
4. Use new generated population for a further run of the algorithm.
5. Stop if the end condition is satisfied and return the best solution in current population else repeat step 2.

3.2 Particle swarm optimization (PSO)

The particle swarm optimization is based on social behavior of flock of birds or school of fish. It is also a population based nature inspired algorithm [5]. The system is initialized firstly in a set of randomly generated potential solutions, and then is performed to search for the optimum one iteratively. It finds the optimum solution by swarms following the best particle. PSO includes elements of exploration and exploitation which helps to avoid the trapping in local optimum position. Some of the key advantages are that this method does not need the calculation of derivatives that the knowledge of good solutions is retained by all particles and those particles in the swarm share information between them. PSO is less sensitive to the nature of the objective function, can be used for stochastic objective functions and is less likely to get stuck in local minima.

Algorithm steps are as follows:

1. Initialize a population of particles, initial searching points and velocities randomly within the allowable range. The current searching point is set to *pbest* for each agent. The *pbest* with best evaluated value is set to *gbest* and the agent number with the best value is stored.
2. Evaluate the objective function value for each agent. If the fitness value is better than current pbest of the agent, the pbest value is replaced by the current value. If the best value of pbest is better than the current gbest, gbest is replaced by the best value and the agent number with the best value is stored.
3. The current searching point is changed according to following equations:

$$v_i^{k+1} = wv_i^k + c_1 rand_1 \times (pbest_i - s_i^k) + c_2 rand_2 \times (gbest - s_i^k)$$

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} \times iter$$

$$s_i^{k+1} = s_i^k + v_i^{k+1}$$

where v_i^k is velocity of agent *I* at iteration *k*, *w* is the weighting function, c_j is weighting coefficients, *rand* is random number, s_i^k is current position of agent *i* at iteration *k*, $pbest_i$ is pbest of agent *i*, *gbest* is gbest of the group, w_{max} is initial weight, w_{min} is final weight, $iter_{max}$ is maximum iteration number and *iter* is current iteration number.

4. Stop if the exit condition is met else proceeds to step 2.

3.3 Artificial bee colony (ABC) algorithm

The artificial bee colony (ABC) algorithm is based on the foraging behavior of bees [6]. The minimal model of forage selection that leads to the emergence of collective intelligence of honey bee swarms consists of three essential components: food sources, employed foragers and unemployed foragers, and the model defines two leading modes of the behavior: the recruitment to a rich nectar source and the abandonment of a poor source.

The value of a food source depends on many factors such as its proximity to the nest, its richness or concentration of its energy, and the ease of extracting this energy. The employed foragers are associated with a particular food source which they are currently exploiting or are "employed" at. They carry with them information about this particular source to the hive and the information can be the distance and direction from the nest, the profitability of the source and share this information with a certain probability. The unemployed foragers are continually at look out for a food source to exploit. There are two types of unemployed foragers: scouts, searching the environment surrounding the nest for new food sources and onlookers waiting in the nest and establishing a food source through the information shared by employed foragers.

3.4 Ant colony optimization (ACO) algorithm

In ACO, an artificial ant is an agent which moves from city to city on a TSP graph. It chooses the city to move to using a probabilistic function both of trail accumulated on edges and of a heuristic value, which was chosen here to be a function of the edges length [7, 8]. Artificial ants probabilistically prefer cities that are connected by edges with a lot of pheromone trail and which are close-by. Initially, m artificial ants are placed on randomly selected cities. At each time step they move to new cities and modify the pheromone trail on the edges used, this is termed local trail updating. When all the ants have completed a tour the ant that made the shortest tour modifies the edges belonging to its tour termed global trail updating by adding an amount of pheromone trail that is inversely proportional to the tour length. Algorithm steps are as follows:

1. Generate randomly a set (C) of possible solutions, set (L) of possible connections among the elements of C. A graph is constructed which is completely connected and weighted where vertices are the components C, set L fully connects the components C, and T is a vector whose components representing the pheromone trail strength.
2. Initialize the pheromone trail strength for all the edges.
3. Set the number of ants in a colony as *m*, and put each ant on a randomly chosen vertex of the graph.
4. All the ants construct their feasible paths by moving to the next vertex based on a probabilistic decision according to state transition rule.

$$p_k(i,j) = \begin{cases} \dfrac{[\tau(i,j)]^{\alpha} \cdot [\eta(i,j)]^{\beta}}{\sum_1^t [\tau(i,u)]^{\alpha} \cdot [\eta(i,u)]^{\beta}} & j,u \in N_{k,i} \\ 0 & otherwise \end{cases}$$

where $\tau(i,j)$ represents the pheromone trail associated with $l_{i,j}$, which is connected between vertices I and j, $\eta(i,j)$ is desirability of adding connection $l_{i,j}$ to the solution under construction

5. The solution construction phase is repeated until all ants have completed their feasible paths.
6. The global updating rule is applied which enforce two things: pheromone evaporation (stops pheromone from unlimited accumulation) and pheromone reinforcement (more pheromone on favorable edges).

   $$\tau(i,j) \longleftarrow (1-\alpha) \cdot \tau(i,j) + \sum_{k=1}^{m} \Delta\tau_k(i,j)$$

   where α is a pheromone decay parameter lying between 0 and 1.
7. Check the termination criterion else return to step 4.

3.5 Honey Bees Mating Optimization (HBMO) algorithm

Recently, honey bees are among the social insects that are the most studied. Honey bees mating optimization (HBMO), one of the search algorithms inspired by the real courtship process of honey bees, is a new swarm intelligence optimization method that is used for simulating social systems [9]. It is a meta-heuristic method inspired by the social phylogenetic of honey bees, be used to solve integrated optimization problems of probability and applied this method to a number of suggestive satisfaction problems. Social insects demonstrate several interesting behaviors: division of labor, individual and group communication and association. These behaviors are due to a combination of the honey bees' genes, nest conditions and ecological environment. Studies on honey bees have revealed much information on molecular genetic problems and the complicated area of socio-genetics. The male haploid structure enables a unique genetic analysis based on the presence of haploid/diploid individuals. New swarm intelligence algorithms based on the haploid and diploid genetic developing operations known as BMO in honey bees have been created and improved for combinatorial optimization problem solutions.

In the honey bees mating optimization algorithm, the procedure of mating of the queen with the drones is described. First, the queen is flying randomly in the air and, based on her speed and her energy, if she meets a drone then there is a possibility to mate with him. Even if the queen mates with the drone, she does not create directly a brood but stores the genotype of the drone in her spermatheca and the brood is created only when the mating flight has been completed. With the term genotype we mean some of the basic characteristics of the drones, i.e. part of the solution. A crossover operator is used in order to create the broods. In a hive, the role of the workers is simply the brood care (i.e. to feed them with the ''royal jelly") and, thus, they are only a local search phase in the Honey Bees Mating Optimization algorithm. Thus, this algorithm combines both the mating process of the queen and one part of the foraging behavior of the honey bees inside the hive. If a brood is better (fittest) than the queen, then this brood replaces the queen.

3.6 Hybridized Honey Bees Mating Optimization (HHBMO) algorithm

The hybrid HBMO algorithm inherits the basic characteristics of the HBMO and incorporates the special features of various algorithms to increase the efficiency of the hybridized algorithm. More specifically, the proposed algorithm uses:

1. The Multiple Phase Neighborhood Search-Greedy Randomized Adaptive Search Procedure (MPNS-GRASP) for the calculation of the initial population of bees and of the initial queen. This procedure is used in order to have a more competitive queen [10].
2. The Expanding Neighborhood Search (ENS) as a local search strategy in order to have more effective and different workers. By using ENS, each brood has the possibility to select randomly the number of workers (local search phases) that will be used for the improvement of its solution.
3. A new crossover operator based on an Adaptive Memory Procedure and on a uniform crossover operator in order to have fittest broods. This crossover operator combines the genotype of the queen and of more than one drones to produce a brood. The reason why such a crossover operator is used is because in real life the queen stores in her spermatheca after the mating the genotype of all drones and after returning to the hive she produces the broods. The adaptive memory procedure is used in order to give the possibility to the queen to store from previous selected good drones (in previous mating flights) part of their solutions, for been able to use them in a new mating flight and for producing fittest broods.

The process of the hybridized HBMO algorithm is briefly given as:

1. Initialize the population of bees using MPNS-GRASP algorithm.
2. Select the best bee among the initial population as the queen. Also, define the maximum number of queen's mating in a single mating flight.
3. Initialize the speed and energy of the queen's mating which are selected at random.
4. The probability of mating of drone with the queen is given as:

$$prob(D) = e^{\left[\frac{-\Delta(f)}{speed(t)}\right]}$$

5. After each transition in space, the queen's speed and energy decay according to the following equations:

$$speed(t+1) = \alpha \times speed(t)$$

$$energy(t+1) = \alpha \times energy(t)$$

6. A brood is generated using the new crossover operator which includes the exploration feature in the algorithm.

4. CONCLUSIONS

The travelling salesman problem is one of the important combinatorial optimization problems in graph theory. The different algorithms used by the researchers to find the optimal path have been reviewed and presented in this article. It is found that the hybrid HBMO algorithm includes the best features of other algorithms which make the algorithm faster and suitable for large-sized optimization problems.

REFERENCES

[1] Gutin, G., Punnen, A., *The Traveling Salesman Problem and its Variations*. Kluwer Academic Publishers, Dordrecht, 2002.

[2] Rastogi, A., Shrivastava, A.K., Payal, N. and Singh, R., "A Proposed Solution to Travelling Salesman Problem using Branch and Bound", *International Journal of Computer Applications*, 65, 2013, pp. 0975-8887.

[3] Boddy, M., "Anytime Problem solving using Dynamic Programming", *AAAI-91 Proceedings*, 1991, pp. 738-743.

[4] Baralia, R., Hildago, J.I., Perego, R., "A hybrid heuristic for the traveling salesman problem", *IEEE Transactions on Evolutionary Computation*, 5, 2001, pp. 1–41.

[5] Goldbarg, E.F.G., Souza, G.R., Goldbarg, M.C., "Particle swarm optimization for the traveling salesman problem", *in: EVOCOP 2006, LNCS, 200*6, pp. 99–110.

[6] Baykasoglu, A., Ozbakor, L., Tapkan, P., "Artificial bee colony algorithm and its application to generalized assignment problem", *in: F.T.S. Chan, M.K. Tiwari (Eds.), Swarm Intelligence, Focus on Ant and Particle Swarm Optimization*, I-Tech Education and Publishing, 2007, pp. 113–144.

[7] Dorigo, M., Gambardella, L.M., "Ant colonies for the traveling salesman problem", *Biosystems*, 43, 1997, pp. 73–81.

[8] Chu, S.C., Roddick, J.F., Pan, J.S., "Ant colony system with communication strategies", *Information Sciences*, 167(1–4), 2004, pp. 63–76.

[9] Fathian, M., Amiri, B., Maroosi, A., "Application of honey bee mating optimization algorithm on clustering" *Applied Mathematics and Computation*, 190, 2007, pp. 1502–1513.

[10] Marinakis, Y., Marinaki, M., Dounias, G., "Honey bees mating optimization algorithm for the Euclidean traveling salesman problem", *Information Sciences*, 181(20), 2011, pp. 4684-4698.