ISSN No. 0976-5697

Volume 8, No. 5, May – June 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Meta-Heuristic Approaches for Solving Travelling Salesman Problem

Elham Damghanijazi
Dept. of Computer Engineering
Islamic Azad University of Aliabad Katool Branch
Aliabad Katool, Iran

Arash Mazidi
Dept. of Computer Engineering
Islamic Azad University of Aliabad Katool Branch
Aliabad Katool, Iran

Abstract: Travelling Salesman Problem (TSP) is one of the problems which is being widely used in transportation industry which its optimization would speed up services and increase customer satisfaction. In this paper, first, TSP is optimized using dynamic programming for Iran59 dataset and a dataset including 10 cities of Iran. Then this problem is solved using 5 meta-heuristic algorithms including Hill Climbing, Simulated Annealing, PSO, Ant Colony and Genetic Algorithm and performance of these algorithms is compared with the optimized solution. Comparison is performed in terms of optimal solution, execution time, and memory. Results show that simulated annealing and hill climbing stop at the local minimum and the proposed tour is longer than other methods. But other algorithms result in better solutions and GA achieves the optimal solution. Comparing the algorithms in terms of execution time shows that GA achieves the optimal solution in shortest time. Moreover, hill climbing method has the lowest memory consumption.

Keywords: Travelling salesman problem; Meta-heuristic algorithms; Genetic algorithm; Dynamic programming

I. INTRODUCTION

Travelling salesman problem was proposed by mathematicians, Carl Menger and Hustler Wietni in 1930 [1]. The problem is that a travelling salesman wants to visit a large number of cities and his goal is to find the shortest path; such that it passes all cities and each city is only passes once and finally returns to the starting point. An example of this problem is shown in Figure 1. In first part of Figure 1, there are 40 points which show cities and in part "B", the optimal path which the salesman should pass to visit all cities is represented.

TSP is one an NP-hard problem [1]. Complexity order of these problems is exponential which does not have an acceptable execution time. In solving such problems, obtaining certain solutions might require much more time than lifetime of the system.

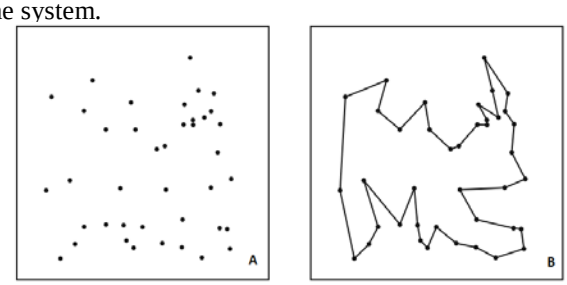

Figure 1. An example of TSP: a) location of 40 cities. B) Optimal path for travelling salesman

Dynamic programming solutions are one of the best methods for solving such terms in terms of execution time and convert time order of the problem into polynomial form. Problem of dynamic programming problems is their memory consumption where in large problems, system cannot meet dynamic programming requirements.

Size and complexity of optimization problems like travelling salesman and real world problems, have attracted attentions of researchers towards meta-heuristic algorithms and local investigation to inspire from social intelligence of creatures [2]. Meta-heuristic algorithms try to obtain logical results in an acceptable time by consuming minimum memory. For example, approaches based on biology and social sciences like evolutionary algorithms, neural networks, swarm intelligence algorithms, genetic algorithm and etc have shown that they provide suitable solutions for different optimization problems.

In this paper, dynamic programming method is used for solving TSP using different heuristic methods like GA, Hill Climbing, PSO, Ant colony and Simulated Annealing and the results are compared. Different algorithms are performed on Iran59 dataset including 59 cities and a smaller dataset including 10 cities of Iran and the results are compared.

The rest of this paper is organized as follows. Section 2 investigates previous works and related algorithms. Section 3 introduces heuristic algorithms, dynamic programming. The implementing and evaluating results obtained from algorithms are in section 4. Finally, section 5 concludes the paper.

II. RELATED WORKS

Several algorithms have been proposed to solve TSP. most of these algorithms are meta-heuristic algorithms and compute the approximate solution in a very short time. Travelling salesman problem is very similar to routing vehicles. Several algorithms are proposed to solve vehicle routing problem which can be used to solve TSP.

Mazidi et.al. have combined GA and Ant colony to solve vehicle routing problem. They have used ant colony to construct the initial population of the GA and have obtained better results compared to meta-heuristic algorithms like PSO [3].

Joshi et.al. have used ant colony algorithm to solve TSP. they have used 2-opt methods for local search and roulette wheel to select the next city. Results have shown that the proposed algorithm determines optimal paths among thousands of cities in shortest time with least cost [4].

Kanthavel et.al. have proposed PSO to solve vehicle routing problem. In this method, two nested optimization algorithms are used and the obtained results show that the proposed method performs better compared to other methods [5].

Sanchez et.al. have solved TSP using GA on a GPU. The proposed method is implemented on GPU in parallel [6].

Tan et.al. have solved vehicle routing problem through combining ant colony and heuristic methods which improve route. In the proposed method, an evaporation function is proposed for pheromones left by ants according to the found route which improves searching optimal paths by the ants [7].

Lei et.al. have investigated vehicle routing problem with time constraint for delivering to each customer and random capacities and an algorithm is proposed for heuristic exploration depending on neighbor is proposed for routing[8].

Urrutia et.al, in their paper have used stack data structure and dynamic programming to solve TSP. results show that optimal solutions are obtained [9].

Marinakis et.al. have combined GA and PSO to obtain better results. In this paper, PSO principles are used to progress solution of GA. This results in better choices in selecting parents of GA and increases exploration in problem space [10].

Mazzeo et.al. have implemented ant colony for vehicle routing with capacity constraint using meta-heuristic algorithms and the results are compared with other algorithms [11].

Yu et.al. have proposed improved ant colony. Main idea of this paper is to increase pheromone. Pheromones increase according to weight of the path which the ant has passed [12].

III. ALGORITHMS FOR SOLVING TRAVELLING SALESMAN PROBLEM

In this section, heuristic methods and dynamic programming method used to solve TSP are presented. In all methods, a vector including cities (no city is visited twice) where starting city is equal to final city is used and sequence of cities is represented. A solution can be seen in Equation (1).

$$Solution = vector[\ City_0, City_1, \dots, City_{n-1}, City_0] \quad (1)$$

Purpose of solving this problem is to minimize the path passed by the salesman which can be seen in Equation (2).

$$Cost = Min(\sum_{i=0}^{i=n-2} Distance(vector[i], vector[i+1]) + Distance(vector[n-1], vector[0])) \quad (2)$$

A. *Dynamic Programming Algorithm*

One of the most widely used methods in designing algorithms is dynamic programming method. This method works on sub-problems like division method and division-based methods. When sub-problems do not overlap, the method performs well. Dynamic programming is widely used in optimization problems. Basic condition for using the method for calculating optimal case is known as optimality principle.

Optimality principle is to solve the problem optimally including optimal solution of all sub-problems. In other words, the problem should be such that upon finding its optimal solution, optimal solution of all sub-problems is also obtained. For example, in finding shortest path between two cities, path between origin and each node on the optimal path is the most optimal path between those cities [13].

In this paper, this algorithm is used to solve TSP. considering optimality principle and dynamic programming, it should be noted which sub-problem is suitable for this problem? It is assumed that we have started from city 1 and a few number of cities have been visited and now we are in city j. Best city should be selected considering that it has not been visited previously.

For a subset of cities $S \subseteq \{1,2,\dots,n\}$, length of shortest path visited at S is shown with C(S,j) which has started from city 1 and has ended at city j. In this method, $C(S,1) = \infty$ because the path could not have started from city 1 and ended at city 1. Now the subproblems should be developed to achieve the main problem. After visiting city j, city i should be visited which is calculated based on length function represented in Equation 3.

$$C(S,j) = MIN(C(S,j) + d_{ij})_{i \in S, i \neq j} \quad (3)$$

In this equation, C(S,j) is size length of the path between city 1 and city j and dij is the length of path between city i and j. Figure 2 shows Psudo-code of the dynamic programming algorithm for TSP.

1. C({1},1) = 0
2. for s = 2 to n
3. for all subsets S ⊆ {1,2,...,n} of size s and containing 1
4. C(S,1) = ∞
5. for all $j \in S, j \neq 1$
6. C(S, j) = $min\{C(S-\{j\},i)+dij : i \in S, i \neq j\}$
7. return $minjC(\{1,...,n\},j)+dj1$

Figure 2. Psudo code of the dynamic programming

The number of sub-problems in this method is 2^n*n and each sub-problem can be solved in a time in linear order. Therefore, cost of this method is equal to $O(n^2 \times 2^n)$.

B. *Hill Climbing Algorithm*

Hill climbing algorithm is one of the simplest local search algorithms. This algorithm starts from a random point in the search space as the initial solution and calculates the objective function. In the next step, neighbors of the initial solution are investigated. If a neighbor with lower value of objective function exists, it changes its location to that point and if there is no better neighbor, current location is selected as the optimal solution.

In this algorithm, search tree is not stored, thus data structure of the current node should only store state and value of the objective function. Hill climbing looks at current neighbors. Hill climbing is sometimes called Greedy local search because it selects a good neighbor without thinking about where to go. Hill climbing usually converges fast because it can improve bad state [14].

In order to solve TSP using hill climbing, it should be noted that due to inherent problems, this method might give local results which are different from real result, however this method can be used to obtain an acceptable result fast. Most important issue in hill climbing is to select the objective function which is considered for the travelling salesman through the passed route. It is obvious that shorter lengths are closer to the final result. Hill climbing algorithm used to solve TSP is shown in Figure 3.

```
1.  HillClimbing(Path: sequence of points)
2.  Begin
3.   Compute the initial length of Path
4.   Loop
5.   Choose the pair of points U,V such that
6.   swapping U with V in Path has the shortest length;
7.   if there is no improvement, then return Path;
8.   swap U and V in the path and decrement the length
     by the change
9.   End of Loop
10.  End
```

Figure 3. Hill Climbing Algorithm [3]

In this algorithm, memory is consumed for the current solution and the neighboring solution. As mentioned, size of each solution is equal to number of cities in the problem. Therefore it requires 2n memory where n is the number of cities.

C. Simulated Annealing Algorithm

This algorithm is inspired from MonteCarlo model (which is a relation between atomic structure, Entropy and temperature during annealing of a material) and it is shown in Figure 4. In this model, first temperature of the material is very high and it is lowered gradually to reach the balance temperature. It is very important that how fast is this temperature lowered and if it is lowered very quickly, material does not reach balance and faces some problems [15].

The idea of minimizing temperature is used to optimize TSP. in implementing this algorithm, first, a solution is created randomly [16].

```
1.  Procedure Simulated Annealing
2.  begin
3.      t ← 0, initialize T
4.      select a current point Vc at random and evaluate Vc
5.      Repeat
6.          Repeat
7.            select Vn from the neighborhood of Vc
8.            if cost(Vc) < cost(Vn) then
9.                Vc←Vn
10.           else if random[0,1) <  e^((cost(Vn) – cost
    (Vc))/T)
11.               Vc ←Vn
12.         until( termination-condition)
13.         T ← g(T)
14.     until (halting-criterion)
15. End
```

Figure 4. Simulated Annealing [3]

In the next step, a neighbor of the initial solution is required for which local search is used. Using the probability in line 10 of the algorithm, neighboring solution is considered as the current solution, even if it does not improve the objective function. Thus, chance of being trapped in local minimum decreases. As time passes. Probability of accepting a solution which does not improve the objective function becomes lower. This process continues until stopping condition is satisfied. In this algorithm, a function called g(T) is used. This function is represented in Equation (4).

$$g(T) = \frac{T_0 - T_F}{Max\ Iteration} \quad (4)$$

In this equation, Max Iteration is number of iterations, where in this paper is 5000, T0 is the initial temperature and Tf is the final temperature which are 1000 and 1 respectively. In this algorithm, like hill climbing algorithm, memory is required for the current solution and the neighboring solution. Therefore, 2n memory is required where n is the number of cities.

D. Genetic Algorithm

main idea of genetic algorithm is to transfer inherited characteristics by the genes. In this algorithm, solutions are known as chromosomes. Procedures of this algorithm include mutation (random changes in chromosomes) and crossover for combining two chromosomes in order to generate a new chromosome. This algorithm is shown in Figure 5.

```
1.  Procedure Genetic
2.  Begin
3.  Choose initial population
4.  Repeat
5.  Evaluate the individual cost of the population
6.  Select pairs of individuals to reproduce
7.  Apply crossover operator
8.  Apply mutation operator
9.  until terminating condition
10. End
```

Figure 5. Genetic Algorithm [3]

Genetic algorithm is a population based algorithm which is performed on a population of solutions. First, 100 solutions are created randomly and route of each solution is obtained using (2). Then parents are selected using several mechanisms. Selecting two superior solutions randomly among 5 superior solutions is the mechanism adopted to select parents. After selecting parents, two solutions are combined by crossover.

In crossover, first one of the parents is selected with equal probability and its initial city is considered as the initial city of the child and another solution is selected between two solution with equal probability and the next city is transferred to the child. This operation continues until child solution is formed.

Mutation is performed considering probability of the created chromosome. Mutation is considered as displacement of two cities in a solution. Algorithm continues until stopping condition is met. At each step, best paths are determined and finally after finishing the algorithm, best path with least cost is explored. In GA, memory is required considering size of population. If population size is m and memory consumed for each solution is n, memory of this algorithm is m*n.

E. Partial Swarm Optimization Algorithm

PSO is one of the most intelligent algorithms in swarm intelligence area. This algorithm is inspired from social behavior of animals like fishes and birds (which live together in small or large groups) and was introduced by James Kennedy and Russel Aberhurt in 1995. In PSO, members of the population are connected directly and interact through exchanging information and remembering memories. PSO is suitable for solving a wide range of continuous and discrete problems and has presented a lot of suitable solution is wide range of optimization problems. Each solution is the search space is a bird which is called a particle. PSO is first initialized by particles which are formed randomly and then it is repeated to search for the optimal solution. In each iteration, each particle changes its future location according to its best

location and best location of the population [17]. Psudo-code of solving TSP using PSO is shown in Figure 6.

```
1.  Procedure Particle Swarm
2.  Begin
3.     For each particle
4.        Initialize particle
5.     For each particle
6.        Calculate cost
7.        If the cost is better than the best cost (pbest)
    in history
8.           Set current value as the new pbest
9.        Choose particle with best cost of all the
    particles as gbest
10.       Update particle position
11.    While maximum iterations or optimum result
12. End
```

Figure 6. PSO algorithm

F. Ant Colony Algorithm

Ant colony algorithm is a tool for solving TSP which was proposed by Durrigo et al in 1992. This algorithm which is an example of multi-operator systems is inspired from food finding behavior of real ants, where each operator is an artificial ant [18,19]. In general, in order to solve TSP using ant colony, 4 steps are required including: 1) initializing parameters including number of ants m, pheromone α, heuristic function operator β, pheromone evaporation ρ, amount of pheromone, maximum iteration. 2) construction solution space; that is each ant is located at several starting points, cities to be visited are computed and it continues until all cities are visited. 3) Updating pheromone; length of each path through which ants pass and record of the optimal solution in the current counter is calculated. 4) Determining stopping condition; determining whether number of iterations has reached its maximum or not. If it has not, one unit is added to counter and associated record is erased and returns to constructing the solution space. Otherwise, it is stopped. Output is the optimal solution. Psudo-code of this procedure is shown in Figure 7.

```
1.  Procedure Ant Colony
2.  Begin
3.     Set Parameter, initialize pheromone trails
4.     While termination condition not met Do
5.        Construct Ant Solution
6.        Update Pheromones
7.     End While
8.  End
```

Figure 7. Ant Colony Algorithm

IV. IMPELEMENTATION AND RESULT

All algorithms are implemented using C#. test data are extracted from two dataset including 59 and 10 cities, respectively. Results of the offered length for both datasets are shown in Figures 8 and 9. As can be seen, path length in GA is the same as length of the optimal path. Results obtained for different algorithms, is the average of 10 times executing the algorithm in 60s on test data.

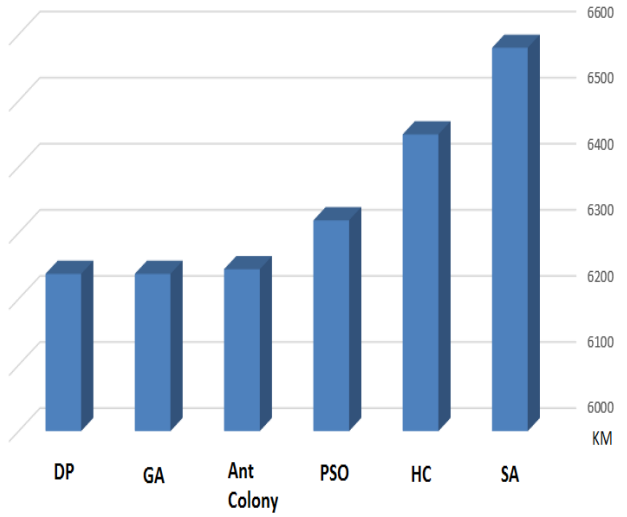

Figure 8. Comparing length of paths obtained from different algorithms in the dataset including 10 cities

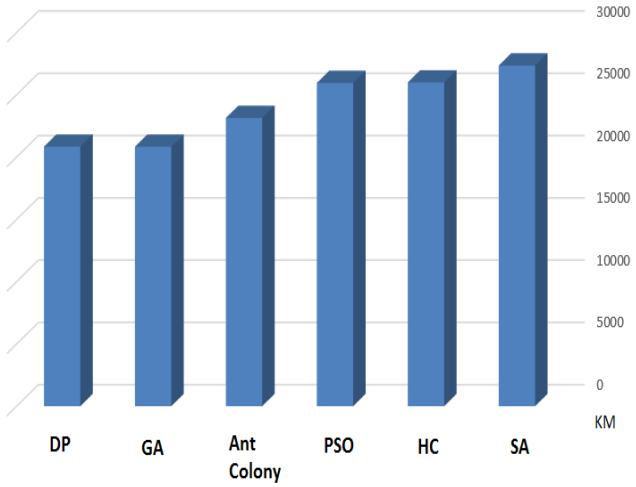

Figure 9. Comparing length of paths obtained from different algorithms in Iran59 dataset

In implementations, two factors are very important to achieve better results. Searching in the problem space (exploration) and searching around the intent solution (extraction) is done locally. The more these factors are balanced, better results are obtained. In simulated annealing algorithm, first a high temperature is considered and exploration is large but as temperature decreases, extraction becomes more. Therefore, first T is considered to be high and a cooling function is designed, which is used to lower the temperature gradually to obtain a high extraction. In hill climbing algorithm, since T is constant, it is selected empirically so that it has both extraction and exploration factors relatively. In GA, crossover operation is used to increase exploration and mutation operation looks for a better solution by performing local search on a solution, thus it performs extraction. In the ant colony algorithm, parameters α and β control extraction of the algorithm which are empirically selected as 1 and 2, respectively. In PSO, by changing weight of matrices, exploration and extraction are controlled. As can be seen in the results, GA and dynamic programming have given the best results but considering computation time, dynamic programming is not suitable and computation time of GA is better than other methods. Figures 10 and 11 show execution time of different algorithms for the two mentioned datasets.

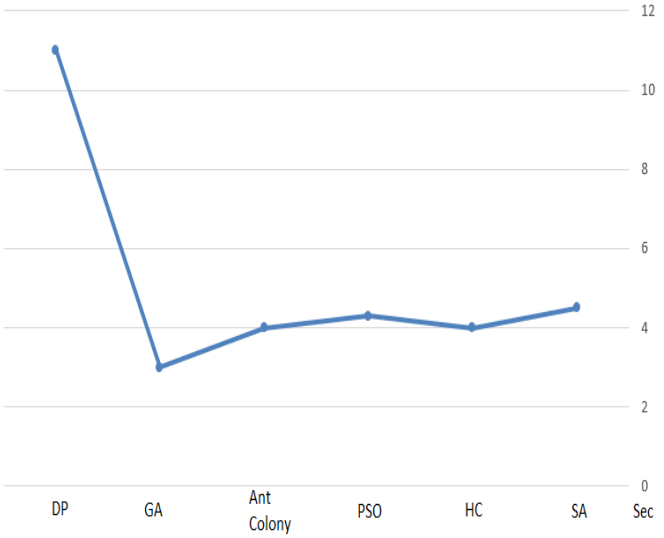

Figure 10. Comparing computation time in the dataset including 10 cities.

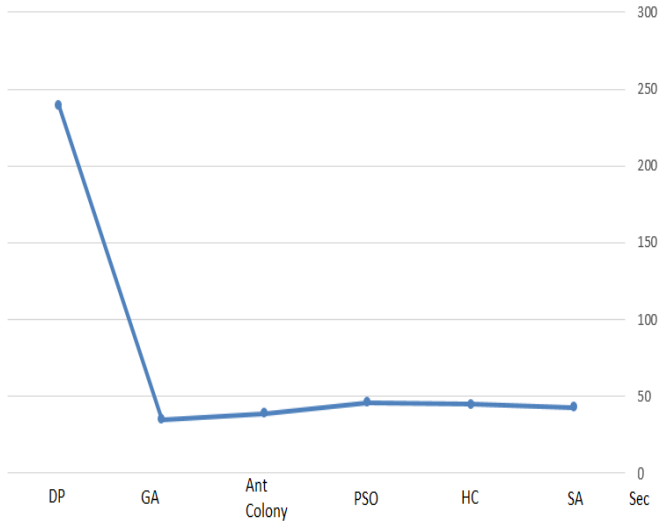

Figure 11. Comparing computation time in Iran59

V. CONCLUSION

In this paper we investigated and compared performance of 5 heuristic algorithms and dynamic programming for solving travelling salesman problem. Dynamic programming is an accurate method, by heuristic methods are approximate which obtain the result in a shorter time. Considering that TSP is an NP-hard problem, using dynamic programming is time consuming and heuristic algorithms are used to obtain the optimal solution in a short time. Results of implementing 5 algorithms and dynamic programming shows that GA gives the best results. In terms of space required for memory, hill climbing has the least memory consumption. Therefore, in this paper, GA is proposed as the best algorithm for solving TSP.
In future studies, performance of other algorithms like gravity search for solving TSP can be studied. In order to guarantee the results, similar experiments are performed on other standard data.

VI. REFERENCES

[1]. D. L. Applegate., R. E. Bixby, V. Chvatal, and W. J. Cook. *The traveling salesman problem: a computational study*. Princeton university press, 2011.

[2]. Giagkiozis, R. C. Purshouse, and P. J. Fleming. "An overview of population-based algorithms for multi-objective optimisation." *International Journal of Systems Science*, Vol. 46, No. 9 pp. 1572-1599, 2015.

[3]. A. Mazidi, M. Fakhrahmad, and M. Sadreddini. "A Meta-heuristic Approach to CVRP Problem: Local Search Optimization Based on GA and Ant Colony." *Journal of Advances in Computer Research*, Vol. 7, No. 1, pp. 1-22, 2016.

[4]. S. Joshi and S. Kaur. "Ant Colony Optimization Meta-heuristic for Solving Real Travelling Salesman Problem." In *Emerging Research in Computing, Information, Communication and Applications*, pp. 55-63. Springer Singapore, 2016.

[5]. K. Kanthavel and P. Prasad. "Optimization of Capacitated Vehicle Routing Problem by Nested Particle Swarm Optimization". *American Journal of Applied Sciences, Vol. 8, No.* 2, pp. 107-112, 2011.

[6]. L. Sánchez, "Parallel Genetic Algorithms on a GPU to Solve the Travelling Salesman Problem." *Revista en Ingeniería y Tecnología, UAZ* , Vol.8, No. 2 , 2015.

[7]. W. F. Tan, L.S. Lee, Z.A. Majidi and H. W. Seow, "Ant Colony Optimization for Capacitated Vehicle Routing Problem.", *Journal of Computer Science, Vol. 8, No.* 6, pp. 846-852, 2012.

[8]. H. Lei, G. Laporte and B. Guo, "The Capacitated Vehicle Routing Problem with Stochastic Demands and Time Windows", *Computers & Operations Research, Vol. 38, No.*12, pp. 1775-1783, 2011.

[9]. S. Urrutia, A. Milanés, and A. Løkketangen. "A dynamic programming based local search approach for the double traveling salesman problem with multiple stacks." *International Transactions in Operational Research*, Vol. 22, No. 1, pp. 61-75, 2015.

[10]. Y. Marinakis, and M. Marinaki, "A Hybrid Genetic–Particle Swarm Optimization Algorithm for the Vehicle Routing Problem". *Expert Systems with Applications, Vol. 37, No.* 2, pp. 1446-1455, 2010.

[11]. S. Mazzeo, and I. Loiseau, "An Ant Colony Algorithm for the Capacitated Vehicle Routing.", *Electronic Notes in Discrete Mathematics, Vol. 18*, pp. 181-186, 2004.

[12]. B. Yu, Z.Z. Yang, and B. Yao. "An Improved ant Colony Optimization for Vehicle Routing Problem", *European Journal of Operational Research, Vol. 196, No.* 1, pp. 171-176, 2009.

[13]. V. L. De Matos, A. B. Philpott, and E. C. Finardi. "Improving the performance of stochastic dual dynamic programming." *Journal of Computational and Applied Mathematics* 290, pp. 196-208, 2015.

[14]. Y. Bykov and S. Petrovic. "A Step Counting Hill Climbing Algorithm applied to University Examination Timetabling." *Journal of Scheduling*, pp. 1-14, 2014.

[15]. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi. "Optimization by Simmulated Annealing.", *Science, 220*(4598), pp. 671-680, 1983

[16]. Sh. Zhan, J. Lin, Z. Zhang, and Y. Zhong. "List-Based Simulated Annealing Algorithm for Traveling Salesman Problem." *Computational intelligence and neuroscience* 2016, 2016.

[17]. F. Glover and M. Laguna, Tabu Search, Kluwer Academic Publishers, 1997simulated annealing, Science, Vol. 220, No. 4598, pp. 671–680, (1983).

[18]. M. Yousefi Khoshbakht and A. Zafari, A new ant colony algorithm for solving multiple traveling salesman problem. The 2th Joint Congress on Intelligent and Fuzzy

Systems (ISFS2008), 28-30 .October, Malek-Ashtar University of Technology, Tehran, Iran. (2008).

[19]. Horri, A., Rahmanian, A., & Dastghaibyfard, G. H. (2015). Energy and performance-aware virtual machine consolidation in Cloud computing a two dimensional approach. Turkish Journal of Engineering, 1, 20–35.

[20]. Rahmanian, A., Dastghaibyfard, G., & Tahayori, H. (2017). Penalty-aware and cost-efficient resource management in cloud data centers. International Journal of Communication Systems, 30(8).

[21]. Ghobaei-Arani, M., Shamsi, M., & Rahmanian, A. A. (2017). An efficient approach for improving virtual machine placement in cloud computing environment. Journal of Experimental & Theoretical Artificial Intelligence, 1–23.