Journal of Computer Science 10 (4): 680-688, 2014
ISSN: 1549-3636
© 2014 Science Publications
doi:10.3844/jcssp.2014.680.688 Published Online 10 (4) 2014 (http://www.thescipub.com/jcs.toc)

OPTIMIZING MULTIPLE TRAVELLING SALESMAN PROBLEM CONSIDERING THE ROAD CAPACITY

Ranjana Ponraj and George Amalanathan

Faculty of Computer Science and Engineering, Hindustan Univertsity, Chennai, India

Received 2013-10-07; Revised 2013-11-04; Accepted 2013-12-23

ABSTRACT

The Multiple Travelling Salesman Problems (MTSP) can be used in a wide range of discrete optimization problems. As the solution to this problem has wide applicability in many practical fields, this NP Hard problem highly raises the need for an efficient solution. The problem is determining a set of routes for the salesmen that jointly visit a set of given cities which are facing difficulty because of road congestion. Selection of proper route is based on the road capacity, which is the deciding factor in the opt vehicle usage. The objective of the study is to optimize the vehicle utilization and minimize the time of travel by salesman based on the road capacity. The solution to this problem is achieved in 3 steps; the first step is by assigning addresses to cities by Ad-assignment algorithm. The second step is by assigning cities and vehicles to salesman by Sl-assignment algorithm. The third step is by using Parallel Shortest Path Multiple Salesman (PSPMS) algorithms to obtain the shortest path. The PSPMS algorithm runs in parallel for each salesman. The solutions to the problem are known to possess an exponential time complexity. From the result we observe that PSPMS is one of the best approximate algorithms used to solve MTSP.

Keywords: Multiple Travelling Salesmen, Road Capacity, Shortest Path

1. INTRODUCTION

Multiple Travelling salesman problem is the extension of the well known travelling salesman problem. This can be applied for various optimization problems in research; in genetic engineering to minimize the length of universal string in DNA sequence, in semiconductor manufacturing, to optimize chain in integrated circuits, in space craft to minimize the usage of fuel, in design of global satellite system network and Matai *et al.* (2010) stated in many real world applications like print press scheduling, crew scheduling and school bus routing. MTSP can also be used to solve the problem in road network. In today's road traffic, congestion becomes a major problem. Selection of proper route would make the company to save much fuel. If a vehicle is stuck in a traffic jam, the vehicle travel time increases accordingly, this results in longer waiting time, from the customer point of view stated by Leontiadis *et al.* (2011).

There were several methods used to schedule automated traffic in network of roads with the help of the scheduler to provide time trajectories for all vehicles, which follow the respective vehicle routes and further ensure that no collision or deadlock will result. But the roads in the transportation network need not have the same capacity. Capacity means the width or the broadness of the road. Some roads are broader and some roads are narrow. If the salesman uses the same type of vehicle in all these roads it will result in congestion. The transport analysis issued by authorized office in Indonesia says that lots of factor affects the smooth flow of traffic. One of the main factors is the level of congestion of the road. Based on the road capacity the type of vehicles that can be used in that road without congestion can be decided. The

Corresponding Author: Ranjana Ponraj, Faculty of Computer Science and Engineering, Hindustan Univertsity, Chennai, India

aim of this study is to solve the MTSP problem by considering the road capacity.

2. MULTIPLE TRAVELLING SALESMAN PROBLEMS

The MTSP have 'm' salesmen to visit a set of 'n' cities and each salesman has to start and end at the same depot. In this, each city must be visited exactly once by one salesman named as s_i, i varies from 1 to m. MTSP can also be defined as a problem of finding the 'k' closed circuit paths, given 'n' cities and 'm' sales men, which minimize the sum of the squares of the path lengths. There are several variations in MTSP like single depot and multiple depot problems. In single depot, all the salesmen start and end the tour at the same point. In multiple depot, the salesman need not end at the startingdepot, but can end at any depot, with the restriction that at end of the tour, the number of salesmen in all the depot should be same as that in the beginning as reported by Levin and Yovel (2012) and Yadlapalli *et al.* (2010). MTSP is also classified as symmetric and asymmetric. In Symmetric MTSP the cost of travel from node n_i to n_j is same as the cost of travel from n_j to n_i. The path is bidirectional. This can be represented by an undirected graph. In asymmetric TSP, the cost of travel from n_i to n_j is different from the cost of travel from n_j to n_i. This can be represented by a digraph.

3. RESEARCH GAP AND PROPOSED WORK

The MTSP problem can be solved by converting MTSP to TSP using ACO algorithm, where the shortest path is determined based on ant behavior Hlaing and Khine (2011). Genetic algorithm is a computational intelligence method, a search technique used in computer science to find approximate solutions to combinatorial optimization problems. Genetic algorithm is proven efficient in solving travelling salesman problem as stated by Albayrak and Allahverdi (2011). Generally for solving MTSP, the problem is converted to TSP and then solved. For converting MTSP to TSP, various clustering methods are used. Each cluster is treated as a sub problem of MTSP and solved. The heuristic solution methords are easy to solve as stated by Bashiri and Karimi (2010). In the real world, Vehicle Routing Problem (VRP) often meets road traffic congestion. The congestion itself may be caused by the number of vehicles hence; the traffic volume is increasing and not balanced within the capacity of existing roads. Erfianto *et al*. (2012) stated that Multi Objective Ant Colony System (MOACS) algorithm solves the VRP problem by considering the level of road traffic congestion as an obstacle. In MTSP the road capacity or edge cost and the road traffic congestion are taken into consideration to solve the problem. In the proposed method, the distance between the cities and the road capacities are taken into consideration to solve MTSP problem.

3.1. MTSP with Road Capacity

This deals with some real world problems where there is a need to account for more than one salesman, given a group of n cities and the distance between any two cities. Suppose there is 'm' salesman starting from a city to visit the group of 'n' cities. Finding the nearly equal shortest tour for each salesman such that each city be visited only once by one salesman and each sales man returns to the starting city at last. MTSP was an appropriate model for the problem of bank messenger scheduling, where a crew of messengers pick up deposits at branch banks and returns them to the central office for processing. To facilitate industrial municipalities to meet the needs of multiple user groups and applications with a single infrastructure by means of multiservice mesh platform, a mesh topology can be used. A computer network topology is the physical communication scheme used by connected devices. A mesh topology involves the concept of routes. In Full mesh topology, each node is connected directly to each of the other node. In partial mesh topology some nodes are connected to all the others, but some of them are only connected to nodes with which they exchange the data because it is less expensive and yields less redundancy. Problems of this type can be addressed by MTSP with road capacity.

The distance between two cities is denoted as edge cost or road length. In addition to this, each edge or road has capacities assigned to them. The capacities differ for different roads. The road capacity is decided based on the size of the vehicle that can be used in that road. The capacities are mentioned using level of the road. In this problem different road levels denotes different road capacities. The different levels of the road are shown in the **Fig. 1**. with different thickness. There will be several cities in all the levels. The capacities can be determined with the road level based on the following assumptions:

$$c_1 = m;\ c_2 = c_1\text{-}2;\ c_3 = c_2\text{-}2;\ c_4 = c_3\text{-}2$$

4. PROBLEM DEFINITION

The MTSP is a special case of vehicle routing problem. The MTSP can be extended to many variations as far as the number of depots and the target paths are concerned, it includes a single depot and multiple depots, as well as closed and open paths. A closed path starts and ends at the same depot, whereas an open path does not require returning to the original depot. The study presents a novel method for solving a MTSP which allows salesmen to start from different depots and end their tours at the original depots. Given a set of 'n' nodes and 'm' salesmen located at each depot, the MTSP aims to find M routes for each salesman starting from a set of depots and ending at the original depots, so that each intermediate node is visited exactly once by one salesman and the total cost is minimized. Let G = (V, E, W,C) be a connected undirectedgraph, where V = {v_1,v_2,...,vn} is a set of cities and E = {<v_i,v_j>|v_i,v_j€V, i ≠ j} is an edge set with a non-negative cost matrix W = {wij| the weight of <vi, vj>} and capacity matrix C = {cij ı the capacity of <vi,vj>}.The graph is said to be symmetric if any <vi, vj>∈E satisfies wij = wji. In the study, we only consider symmetric graphs that satisfy the triangle inequality:

- Definition 1: w(v_i, v_j) is the distance between vi and v_j, denoted by w_{ij}
- Definition2: c(v_i,v_j) is the capacitybetween v_i and v_j denoted by c_{ij}
- Definition 3: A tour denotes a route that starts at one node and ends at the same node

The transportation problem shown in **Fig. 1** can be solved by means of PSPMP algorithm. Here, both the edge weight and edge capacity (road capacity) are taken into consideration. Based on the road capacity the salesman is assigned with vehicles. Many exact and approximate algorithms were developed to solve TSP one among that is proposed by Xu *et al*. (2013). The exact algorithms tend to be very time consuming, because their time complexity is super polynomial. An alternative, perhaps more practical approach is to design approximate algorithms which give solutions of reasonable quality in a short time. So the approximate algorithm is taken to solve the problem.

4.1. Problem Formulation

The structure of MTSP is represented as a graph, where the cities are denoted as nodes in a graph. The connection between pair of cities denotes edges in a graph. Goyal (2010) proposed that each edge has a cost associated with it known as the distance between two cities. In addition to the edge cost, the edge capacity is also taken into consideration to solve the problem. Let G = (V, E, W, C) be a connected un direct graph, where V = {v_1,v_2,……..v_n} where n is set of nodesand E is the edges. The weight $w_{i,j}$ is associated with each edge and the capacity $c_{i,j}$ is also associated with the edges. **Table 1** shows the MTSP problem of **Fig. 1** with number of salesmen and cities.

The travelling salesman problem introduced here is considered in a different approach, suppose a company is planning to send salesman a trip to several cities to meet the customers and come back to the city where he started. In this problem, we assume that, level 1 cities the salesman can visit through one mode of transport, say air service, but for level 2 cities, the same mode may not be possible. For level 2 cities, another mode of transport like bus service is possible and in particular level cities only two-wheeler service may be possible. This can be solved by assigning different types of vehicles. One set of salesman uses air service, the other set of salesman uses bus service and the other set of salesman uses two wheeler service based on the road capacity. Let us consider there are 4 levels of roads. **Table 1** shows the number of nodes needed in each level and the number of salesmen needed. The total number of nodes in all the levels are 256. There are nodes 84 common nodes. From this we summarize that the level of the roads determiners the number of salesman. For level 1, roads one salesman is needed, for level 2 roads 4 salesmen are needed, this can be written in a generalized form as:

$$\text{No. of Salesman} = \text{Level} \times 4$$

Table 2 shows the type of vehicle used and the number of vehicles used in each type. The number of vehicles needed depends on the number of salesman. The vehicles are of type vh_1, vh_2, vh_3 and vh_4. The vehicles are differentiated based on the size and capacity of the vehicle.

The number of salesmen is directly proposition to the number of vehicles. This is applicable only for the above example where all the cities are connected to 3 other cities of same level. The problem can be solved in 3 stages:

- Assigning addresses to the cities with Ad-assignment algorithm
- Assigning salesman to cities based on the capacity using SL-assignment algorithm and
- Applying Parallel Shortest Path Multiple Salesman Algorithm (PSPMS) to find the shortest path for each salesman in parallel

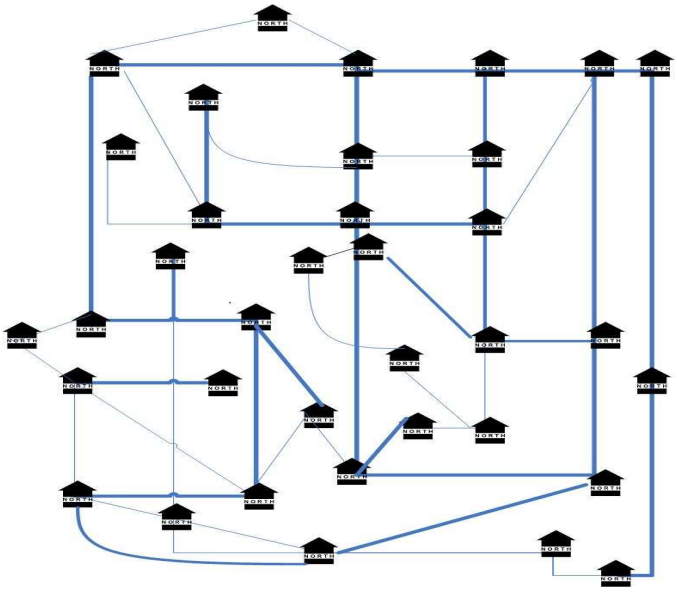

Fig 1. Network showing cities connected by roads of varying capacities

Table 1. Nunumber of nodes and salesmen needed

No. of salesman	Level	No. of nodes	Common nodes	Total No. of nodes
1	1	4	0	4
4	2	04×4 = 16	4	12
16	3	16×4 = 64	16	48
64	4	64×4 = 256	64	192
		84	256	

Table 2. Vehicle allotment to salesmen

No. of salesman	Levels needed	Vehicle types	No. of vehicles
1	1	vh_1	1
4	2	vh_2	4
16	3	vh_3	16
64	4	vh_4	64

5. ASSIGNING ADDRESS TO CITIES

Computing optimal routes in a road network is one of the focuses of real world applications of algorithms. Our bench mark throughout the study is Indian road which has 192 nodes and 256 edges. This can be denoted as a non linear multiple travelling salesman problem as proposed by Nallusamy *et al*. (2010). The input to the algorithm is the adjacency matrix with the weight assigned to each edge i.e., connecting the nodes n_i, n_j.

The graph also has the capacity matrix in addition to the weight matrix. The capacity matrix gives the capacity of edges connecting the pair of nodes. The capacities are assigned based on the level of the road. Level of the road is decided based on properties on the road. The property taken here is the width of the road.

Roads like 6lane, 4lane. In the proposed work we assume road with broadness based on lane. So based on the level of the road the capacities are fixed in the following manner. For example roads are at level l_1, l_2,…ln then the capacity of the node can be any value, which can be calculated with the capacity of first level node. After assigning weight and capacity to edges the next step is to read the capacity and assign address to cities. For example the cities at level 1 are assigned 4 digit integer value having value only in the thousands position. The values in thousands position will be incremented for level 1 nodes. For level 2 nodes the value in hundreds position is increment by 1 each time:

Level 1 node address as 1000,2000,3000,…..
Level 2 nodes address 1100,1200..2100,2200…..
Level 3 nodes address 1110,1120,2110,2210….
Level 4 nodes address 1111, 1121,…,2111,2211

Based on the above sequence the addresses are assigned for all the cities.

The algorithm 1 explains the address assignment to cities.

Algorithm 1: Ad_assignment (Graph, Source)
Input: Graph G = (V, E) with edge-capacity.

1. Initialize A to be empty;
2. Place each vertex in its own set;
3. Sort edges of G in increasing-order;
4. for each (v_i,v_j) in G// take each pair of nodes in the sorted set
5. I = 1;a = 1000; // initialize the first address
6. addr [v_i] = i×a; //Assign address to v_i;
7. increment i; // Increment for assigning next address
8. addr[v_j] = i×a;//Assign address to v_j;
9. increment i; //Increment for assigning next address
10. Else
11. A = a div 10;
12. Add addr[v_i], addr [v_i] to A; //add vi and v_j to address assigned values

The algorithm reads the adjaceny matrix and assigns address for assignment operation it is only 1×1×1….n = 1.

For matrix operation, which is a two dimensional array the complexity is O (n^2).

Thus the addresses are assigned to all the 256 nodes for common nodes if address is already assigned it will be ignored during assignment of address for the next time. All the 256 nodes are assigned address based on their level. All the 256 cities are assigned with a 4 digit address.

6. ASSIGNING CITIES TO SALESMAN

After assigning the address to cities based on the Ad_assignement algorithm. Each salesman is allotted with number of cities to travel. This is done by taking the city addresses. Let it be addr[v_1], addr [v_2]. This will be in an array A. The address of the cities is read from the array. The address values are 4 digit numbers. A constant 'Ar' is fixed with the value of 1000 and the address is divided by 'Ar' the remainder value is assigned to salesman j, where j is initially 1. The process is continued until there is a value for the address if not the 'Ar' is divided by 10 and the process is continued till all the salesmen are assigned with cities. For assigning cities to next salesman the j value is incremented by 1.

The process of assigning cites to salesman is shown in the algorithm 2:

Algorithm 2: SL_Assignement 2 (A, source)
Input: Graph = (V,E) with vertex-address.

1. Initialize S to be empty;
2. Place each vertex address in its own set
3. Sort the vertex of a graph in increasing order
4. for each vertexin the set A(v_i,v_j)//start the loop takingpair of nodes in the sorted list
5. Initialize $_I$ = 0; $_j$ = 0; // initialize i and j values
6. Ar = 1000; // initialize the 'Ar' value as 1000;
7. If vertex_address mod Ar;//divide the node address by m and if remainder is zero.
8. assign to s[i] [j]; //assign the remainder values as salesman number
9. Assign vh[k] to s[i][j] // assign the vehicle number as salesman number
10. j++//increment j
11. Else// if vertex address has remainder
12. Ar=Ar/10; //divide the value of 'Ar' by 10
13. I++;//increment i;
14. End if //end
15. End for //end loop if all salesmen are assigned with cities and vehicles
16. Return S

For example if the city address is 1000, 2000 …, the address is divided by 1000 and there is a remainder values like 1,2,3 so it is assigned to salesman1. 1100, 1200… is the next level so it is divided by 100 and the remainder is 11,12 so assigned to saleman2, but 2100, 2200 n2 since the first digit is not same. The cities address 2100, 2200 … is assigned to salesman3. Thus the algorithm assign salesman with the cities based on the city address.

6.1. Assigning Salesman with Vehicle

If salesmen are assigned with cities, it easy to assign salesman with vehicle. The salesman visiting the cities at level1 is assigned with a vehicle type v1. The salesman travelling at level 2 is assigned with a vehicle of type v3. From **Table 2** we find that the number of vehicle in each type depends on the number of salesman at each level. So vehicles are assigned to salesman, while assigning cities to the salesman. The algorithm 2 shows the allotment of cities and vehicles to salesman. The above SL-assignement have the complexity of O (n^2).

For reading the adjacency matrix with city address it is O (n^2).

For assigning salesman with address the complexity is 1×1×1×…n≈1.

For assigning salesman with vehicle it is again it takes 1×1×1×…n≈1.

So the run time complexity of the SL-Assignement2 algorithm is O (n^2).

7. PARALLEL SHORTEST PATH MULTIPLE SALESMAN ALGORITHM (PSPMS)

The classical way to compute the shortest path between the given nodes in a graph is with the given edge lengths. On this network, Dijkstra's algorithm takes more than a second on a state-of-the-art workstation to compute the shortest path between two random nodes. This is too slow for many applications. To overcome this PSPMS algorithm is used. The aim of the algorithm is to obtain the shortest route in less time. The algorithm will start at level 1 vertex. The salesman assigned to thelevel1 cities starts the tour at the level1 nodes with the broader vehicles named as 'vh1'. Now Salesman s1 will be using the vehicle vh1. The salesman finds the shortest path among the allotted level 1cities using the Shortest path algorithm. The algorithm inputs a graph G (V, E) and salesman set S with s1, s2…sn salesmen. All salesmen are allotted with cities using SL-Assignment algorithm. The salesman 1 starts the tour from the source vertex v1, which can be assumed as depot for level1 cities. This depot will be common for salesmen at different levels. If the salesman visits the next city, all other salesman in that city also takes his tour to find the shortest path. Thus this algorithm runs in parallel.

The **Fig. 2** shows the cities with the node level to decide road capacity. There will be several node or cities known as common node. The common node will have a road to level i and level j of cities. In that common node the salesman will hand over the charge to the other salesman. Because the salesman allotted to level1 cities cannot travel to level2 cities. Next the level2 salesman takes in charge to find the shortest path. Thus the salesman takes his tour in parallel with the assigned vehicle based on the road capacity. Salesman at level1 will have one type of vehicle v1. Salesman at level2 will have another type of vehicle denoted as v2, so if there are 'n' levels of road then the salesman will have 'vn' vehicle types. The PSPMS algorithm continues till the salesman travel through all the cities. Thus there will be parallel execution of the algorithm by different salesman in different levels. By this the salesman can reach the city using shortest route and with optimal assigned vehicle.

Algorithm 3: Parallel shortest path multiple travelling salesman Algorithm PSPMS
Input: Graph G = (V, E, w) with edge weights

1. Let S [i] be the salesman set with I = 1, 2…n. //S be the array of salesman
2. Let p[i] be the processor set with I = 1,2,…n //P be the array of processor
3. The adjacency matrix is portioned by S-vecto //Partition the salesman array
4. For each salesman i of n do in parallel
   //start the loop for each portioned salesman array
5. Read addr[v_i] of s[i]
   //read the city address of salesman
6. Assign the values to the processor P[i]
   //assign to processor i
7. Get theadjacency Matrix offor p[i] denoted by D
   //get the values in then array for P
8. For every vertex v in D
   //read the vertices in D
9. Dist [source] = 0
   //Initial the source vertex distance
10. Q: =set of all node in D //
    put the D vertex set to Q
11. While Q not empty
    //start the loop while Q is not empty
12. U = vertex in Q with smallest distant dist [];//find minimum element from Q named as 'u'
13. Remove u from Q //remove 'u' from the queue
14. for each neighbor v of u.
    //find the adjacent node of 'u'
15. Sp = dist[u] + dist-between (u, v) ;
    //find the shortest path
16. If sp < dist[v]:
17. Dist[v]: = sp;
18. Previous[v]: = u;
19. Decrease-key v in Q;
20. End if
21. End for//end of for loop
22. Node is broadcasted to D-vector //send the result to the D
23. End for in parallel//do in parallel for all i values of p
24. End while //end while
25. Return dist; //end

The algorithm makes use of parallel algorithm technique proposed by Vaira and Kurasova (2011), to find the shortest path. The PSPMS involves portioning the adjacency vertex in D blocks with the allotted salesman to the vertex. Salesman 1 is put in one vertex, salesman2 in other vertex and so on. Each portioned vertex set, based on salesman is put into different processor. Each processor has the node n_i, n_j of same capacity. The node is broad casted into all processor and the D-vector is updated.

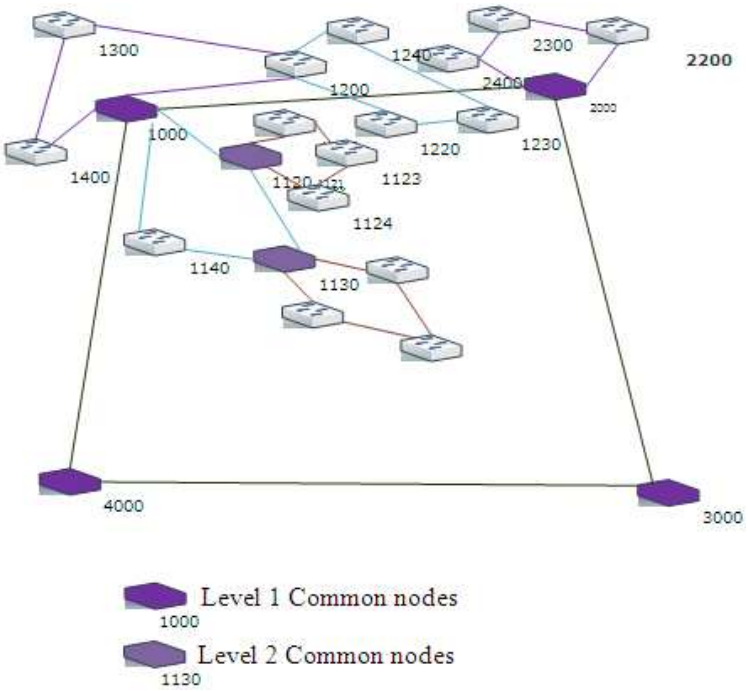

Fig. 2. Cities with levels and road capacity

The source is portioned and the processor finds the single source shortest path from very vi to all other vertices by PSPMS algorithm. It is found the parallel running time is O (n^2). Cost optimal can use 'n' processor where 'n' depends on the problem size. It can use n^2 processor for smaller problem size. If the problem size increases it can use up to n^3 processor in the worst case. If there are p processors, with p>n then it can be executed in n/p processor. So the computation time is $T_p \approx O (n^3/p)$. To broadcast the node to all processor and to update the D-vector the time taken is O (nlog p). So the $T_p \approx O (n^3/p) + O(n \log p)$. Which is less complexity than O (n^3). The above PSPMS algorithm is executed and compared with the sequential algorithms and it shown considerable decrease in the running time.

8. RESULTS

The computational complexity of sequential algorithm is shown **Table 3**. For cities of four levels. The computational complexity of the existing algorithm for more numeber of cities is high in the order of $O(n^3)$. **Table 4** shows the computationl complexity for the proposed PSPMS algorithm. The algorithm runs in parallel for 85 processor. In each processor n = 4, so the time complexity is minimized. Here we use processor of size 85.Thee complexity of using this PSPMS algorithm is $O(n^3/p)$.

Table 3. Existing Algorithm

P	N	Sequential
1	4	64
1	12	1728
1	48	110592
1	192	7529536

Table 4. PSPMS algorithm

P	N	PSPMS
1	4	64
4	12	432
16	48	6912
64	192 117649	

9. DISCUSSION

The computation time for the propose PSPMS algorithm is $T_p \approx O (n^3/p)$, which is less complexity than O (n^3), the existing algorithm. The existing algorithm makes use of one procesor and runs sequentially. **Figure 3** shows the computational complexity of the existing algorithm for each level of cities. The computational complexity is represented in exponential order. **Figure 4** Shows the computaional complexity of the PSPMS algorithm with exponential order.

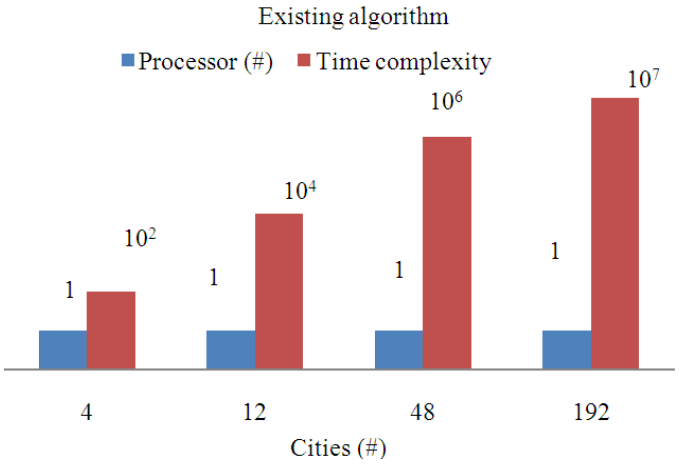

Fig. 3. Performance of the existing algorithm

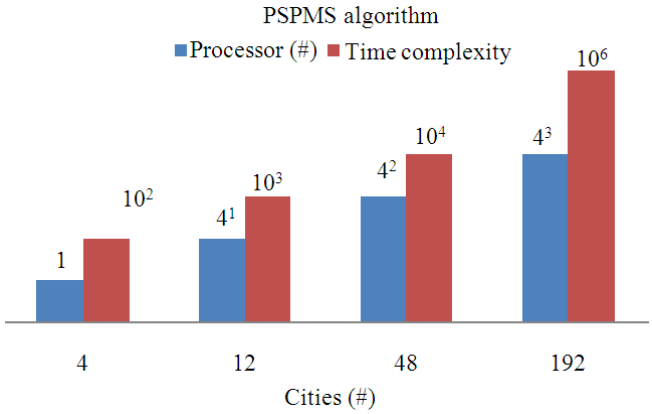

Fig. 4. Performance of PSPMS algorithm

This makes uses of one processor for 4 cities, so for each level of cities the processer increases in the order of 1, 4^1, 4^2, 4^3 totally of 85 processor. Comparing the graph in **Fig. 3 and 4** shows the computational complexity of the proposed PSPMS algorithm is less time consuming.

10. CONCLUSION

The multiple salesman problem is solved by considering the road capacity. The solution to the problem is obtained in 3 states by Assigning address to cities, assigning cities to salesman and finally finding the shortest route for each salesman by executing the algorithm in PSPMS algorithm. The advantage of this algorithm is time complexity reduction. For more number of cities it takes $O(n^3/p)$, which is less than the existing sequential algorithms and very easy to implement .In the proposed work, since the road capacity is also taken into consideration, the salesman need not use the same type of vehicle for all the roads. The salesmen uses different vehicle in different roads. This increases the optimal use of vehicles. The limitation of the approach presented in this study is, workload of the salesman is balaced only if the cities are evely distributed in all levels. This work can be further extended by balancing the workload of the salesmen with varying number of cities at different levels.

11. REFERENCES

Albayrak, M. and N. Allahverdi, 2011. Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms, Expert Syst. Applic., 38: 1313-1320. DOI: 10.1016/j.eswa.2010.07.006

Bashiri, M. and H. Karimi, 2010. An analytical comparison to heuristic and meta-heuristic solution methods for quadratic assignment problem. Proceedings of the 40th International Conference on Computers and Industrial Engineering, Jul. 25-28, IEEE Xplore Press, Awaji, pp: 1-6. DOI: 10.1109/ICCIE.2010.5668262

Erfianto, B., J. Ridha and I. Adiwijaya, 2012. Implementation of vehicle routing problem using multi-objective ant colony system with obstacle. Proceedings of the 1st Taibah University International Conference on Computing and Information Technology, (CIT, 12).

Goyal, S., 2010. A survey on travelling salesman problem. University of North Dakota.

Hlaing, Z.C.S.S. and M.A. Khine, 2011. Solving traveling salesman problem by using improved ant colony optimization algorithm. IJIET, 1: 404-409. DOI: 10.7763/IJIET.2011.V1.67

Leontiadis, I., G. Marfia, D. Mack, G. Pau and C. Mascolo *et al*., 2011. On the effectiveness of an opportunistic traffic management system for vehicular. IEEE Trans. Intell. Trans. Syst., 12: 1537-1548. DOI: 10.1109/TITS.2011.2161469

Levin, A. and U. Yovel, 2012. Local search algorithms for multiple-depot vehicle routing and for multiple traveling salesman problems with proved performance guarantees. J. Combinat. Optim. DOI 10.1007/s10878-012-9580-x

Matai, R., S.P. Singh and M.L. Mittal, 2010. Traveling salesman problem: An overview of applications. formulations and solution approaches. Indian Institute of Technology Delhi.

Nallusamy, R., K. Duraiswamy, R. Dhanalaksmi and P. Parthiban, 2010. Optimization of non-linear multiple traveling salesman problem using k-means clustering, shrink wrap algorithm and meta-heuristics. Int. J. Nonlinear Sci., 9: 171-177.

Vaira, G. and O. Kurasova, 2011. Parallel bidirectional dijkstra's shortest path algorithm. Proceedings of the Conference on Databases and Information Systems, (DIS' 11), IOS Press Amsterdam, pp: 422-435. DOI: 10.3233/978-1-60750-688-1-422.

Xu, L., Z. Xu and D. Xu, 2013. Exact and approximation algorithms for the min-max k-traveling salesmen problem on a tree. Eur. J. Operat. Res. DOI: 10.1016/j.ejor.2012.12.023

Yadlapalli, S., S. Rathinam and S. Darbha, 2010. 3-Approximation algorithm for a two depot, heterogeneous traveling salesman problem. Optim. Lett., 6: 141-152. DOI: 10.1007/s11590-010-0256-0