HÖGSKOLAN
Dalarna

Use Multilevel Graph Partitioning Scheme To Solve Traveling Salesman Problem

Master Thesis Report

Reg No. E3915D

Muhammad Umair Khan

2010

Program **Master of Science in Computer Engineering**	Reg. Number **E3915D**	Extent **15 ECTS**
Name of Student **Muhammad Umair Khan**	Year-Month-Day **2010-05-24**	
Supervisor **Pascal Rebreyend**	Examiner **Professor Mark Dougherty**	
Company None	Supervisor at the company None	
Department **Department Of Computer Engineering, Dalarna University, Sweden.**		
Title: **Use Multilevel Graph Partitioning Scheme To Solve Traveling Salesman Problem**		

ABSTRACT

The traveling salesman problem is although looking very simple problem but it is an important combinatorial problem. In this thesis I have tried to find the shortest distance tour in which each city is visited exactly one time and return to the starting city. I have tried to solve traveling salesman problem using multilevel graph partitioning approach.

Although traveling salesman problem itself very difficult as this problem is belong to the NP-Complete problems but I have tried my best to solve this problem using multilevel graph partitioning it also belong to the NP-Complete problems. I have solved this thesis by using the k-mean partitioning algorithm which divides the problem into multiple partitions and solving each partition separately and its solution is used to improve the overall tour by applying Lin Kernighan algorithm on it. Through all this I got optimal solution which proofs that solving traveling salesman problem through graph partition scheme is good for this NP-Problem and through this we can solved this intractable problem within few minutes.

Keywords: Graph Partitioning Scheme, Traveling Salesman Problem.

ACKNOWLEDGMENT

This thesis work was initiated as a part of the study of International Master Program of Computer Science specialization in Artificial Intelligence at Högskolan Dalarna (Dalarna University) in Sweden.

As the author, I wish to express my appreciation for my dedicated supervisor Pascal Rebreyend. I am able to complete this thesis because of his guidance and his long-term unfailing support. He has not only offered me valuable ideas and suggestions, but also criticisms with his profound knowledge in Traveling Salesman Problem through multi level graph partitioning approach and rich experience in research. His patience and kindness are greatly appreciated. He is always willing to discuss with me anytime he is available.

I am also extremely grateful to Prof. Mark Dougherty, Prof. Jerker, Dr. Hasan Fleyeh and Dr. Yella Siril. I have learnt a lot from them - not only about the knowledge relevant to this thesis, but also the professional ethics. I am very much obliged to their direct and indirect efforts of helping me to complete this thesis.

At last but not least, my deep gratitude goes to my dearest parents, family members for their infinite love.

Muhammad Umair Khan

Table Of Content

ACKNOWLEDGMENT 5

CHAPTER 1 AIM OF THESIS 10

1.1 TRAVELING SALESMANPROBLEM 11

1.2 HOW TO IMPLEMENT A MULTILEVEL GRAPH PARTITIONING SCHEME TO OBTAIN A SOLUTION OF TSP 11

CHAPTER 2 INTRO. TO THE TSP 12

2.1 THE TRAVELING SALESMAN PROBLEM 13

2.2 TYPES OF TRAVELING SALESMAN PROBLEM 14

2.3 P-CLASS 14

2.4 NP-CLASS 15

2.5 PROOF THAT TRAVELING SALESMAN PROBLEM IS NP-COMPLETE 15

2.6 PROBLEM THAT CANNOT BE SOLVED IN POLYNOMIAL TIME 16

2.7 INSTANCES USED IN THIS THESIS 16

CHAPTER 3 HEURISTIC USED FOR SOLVING TSP 17

3.1 TOUR CONSTRUCTION HEURISTIC 18

3.1.1 NEAREST NEIGHBOR 18

3.1.2 RANDOM 19

3.1.2 GREEDY 19

3.2 LOCAL SEARCH HEURISTIC 19

3.2.1 2-OPT 20

3.2.2 3-OPT 20

3.3 ITERATIVE METHODS 20

3.3.1 SIMULATED ANNEALING ALGORITHM 21

3.2.2 LIN KERNIGHAN ALGORITHM 21

CHAPTER 4 MULTILEVEL GRAPH PART. APPROACH 23

4.1 WHAT IS GRAPH PARTITIONING APPROACH.....24

4.2 PSEUDO CODE OF MLS FOR GRAPH PARTITIONING.....25

4.2.1 THE COARSEN STEP.....25

4.2.2 THE INITIALIZE STEP.....26

4.2.3 THE REFINEMENT STEP.....26

CHAPTER 5 HOW TO SOLVE THE TSP.....27

5.1 K-MEAN PARTITIONING.....28

5.2 WHY USED K-MEAN PARTITIONING.....28

5.3 COARSENING STEP BY USING K-MEAN PARTITIONING.....28

5.3.1 HOW TO CALCULATE THE NO.OF PARTITIONS.....30

5.3.2 WORKING OF HOW TO ASSIGN NODES TO PARTITIONS.....30

5.3.3 ITERATION NO.0 HOW TO SELECT CENTROID.....30

5.3.3.1 ITERATION NO.0 HOW TO CALCULATE THE DISTANCE BETWEEN CENTROID AND CITIES.....31

5.3.3.1.1 ITERATION NO.0 DISTANCE BETWEEN CENTROID 1 AND CITIE.....31

5.3.3.1.2 ITERATION NO.0 DISTANCE BETWEEN CENTROID 2 AND CITIES.....31

5.3.3.2 ITERATION NO.0 THE DISTANCE MATRIX REPRESENTING DISTANCE BETWEEN CENTROIDS AND CITIES.....32

5.3.3.3 ITERATION NO.0 THE GROUPING MATRIX.....32

5.3.4 ITERATION NO.1 HOW TO SELECT CENTROID.....32

5.3.4.1 ITERATION NO.1 HOW TO CALCULATE THE DISTANCE BETWEEN CENTROID AND CITIES.....32

5.3.4.1.1 ITERATION NO.1 DISTANCE BETWEEN CENTROID 1 AND CITIES.....33

5.3.4.1.2 ITERATION NO.1 DISTANCE BETWEEN CENTROID 2 AND CITIES.....33

5.3.4.2 ITERATION NO.1 THE DISTANCE MATRIX REPRESENTING DISTANCE BETWEEN CENTROIDS AND CITIES.....33

5.3.4.3 ITERATION NO.1 THE GROUPING MATRIX.....33

5.3.5 ITERATION NO.2 HOW TO SELECT CENTROID.....34

5.3.5.1 ITERATION NO.2 HOW TO CALCULATE THE DISTANCE BETWEEN CENTROID AND CITIES 34
5.3.5.1.1 ITERATION NO.2 DISTANCE BETWEEN CENTROID 1 AND CITIES 34
5.3.5.1.2 ITERATION NO.2 DISTANCE BETWEEN CENTROID 2 AND CITIES 34
5.3.5.2 ITERATION NO.2 THE DISTANCE MATRIX REPRESENTING DISTANCE BETWEEN CENTROIDS AND CITIES 35
5.3.5.3 ITERATION NO.2 THE GROUPING MATRIX 35
5.3.6 WHEN K-MEAN PARTITIONING STOP WORKING 35
5.4 UNCOARSENING PART 36
5.4.1 SOLVED EACH SEPARATE-PARTITIONS 36
5.4.2 LIN KERNIGHAN ALGORITHM 37
5.5 WHY USED LIN KERNIGHAN ALGORITHM 37
5.6 THE EUCLIDEAN DISTANCE 37

CHAPTER 6 EXPERIMENTAL RESULTS 39
6.1 SYSTEM INFORMATION 40
6.3 COMPARISON OF RESULT BY DIVIDING IT MIN. AND MAX PARTITIONING 41
6.4 EXPERIMENT BY APPLYING LKH ON THE INFO OBTAINED BY K.-MEAN PARTITIONING 43

CHAPTER 7 OVERVIEW ON EXPERIMENT 45
7.1 DIVISION OF PROBLEM INTO MULTIPLE PARTITIONS 46
7.2 ADAPTION OF LIN KERNIGHAN ALGORITHM 46
7.3 CONCLUSION 47

REFERENCES 48
TSP INSTANCES PROBLEM 51

Chapter No.1

Aim of Thesis

1.1 Traveling Salesman Problem

The Travelling Salesman Problem (TSP) is one of the most intensively studied problems all round the world. Basically in this problem a salesman need to visits each city one time and returns back to the city from the start point of traveling. This problem is also belonging to the class of "NP-Complete" problems. Means that this problem takes exponential time in their solution as NP-Complete problems are intractable in nature that still nobody can give the efficient way of solving them for large "n".

1.2 How To Implement Multilevel Graph Partitioning Scheme To Obtain A Solution For Traveling Salesman Problem

I have tried to implement a multilevel scheme in such a manner that it can produce optimal solution. As everyone know that both Traveling Salesman Problem and Multilevel Graph Partitioning belong to the class of NP-Complete. I have divided the problem into multiple sub-problems by using the technique of the k-mean partitioning scheme which can divides the problems into multiple partitions solve it each individual partition separately and its solution is used to improve the overall tour by using the Lin Kernighan al the lem one by one to again build the graph. I have used the Lin Kernighan algorithm in the refinement step of the Multilevel Scheme. Because Lin Kernighan is the best heuristic that can further improved the tour by using different local search methods.

Chapter No.2

Introduction to the Traveling Salesman Problem

2.1 The Travelling Salesman Problem

The Travelling Salesman Problem is looking simple but combinatorial problem. In a travelling Salesman Problem a salesman need to visit all "n" cities ("nodes") in a cyclically order. A salesman in his tour need to visit each city one time and finished up when he reached the starting city or node. Many Travelling Salesman Problem are symmetric as the distance between two cities Let suppose "A" and "B" is equivalent whether salesman travel from A to B or B to A. in this case you will get exactly the same tour length if you reverse the order of visited. If there are only 2 cities then this problem is trivial and there solution is tractable as one tour is possible. If we consider 3 cities in symmetric case it is also trivial. If salesman need to travel "n" cities then there are (n-1)! Different tour this is known as asymmetric TSP this problem is also called intractable problem. Pick any city as the first from "n" cities then u have "n-1" choices for the second visited city and "n-2" choices for the third city and so on that's why called these problems are intractable. For the symmetric case there are many distinct solutions (n-1)! /2 for an "n" city TSP. In both case Symmetric case and also in Asymmetric TSP the number of solutions becomes extremely large for large value of "n" cities. So it is very difficult to solve this problem on a single processor because it might be possible that it takes a month or year to try to solve these types of problems.

Most of the persons shows their interest to solve this problem is due the tsp problem belong to the NP class problems. In NP class there is some problems which solves in polynomial time and non polynomial time. If anyone solved the solution of problem of NP in polynomial time then we will easily got the solution of other non polynomial time problems. So everyone is trying to proof that P=NP or NP=P.

2.2 Types Of TSP Problems

According to the research on the TSP we have categorizes the problem of TSP into three types. The type of problem depends on how the input function is given. The names of functions are

1. Euclidian symmetric
2. Asymmetric
3. Random distance matrixes

In the term Euclidian the problem is given in the 2-D plane. We can calculate the distance between two points through the Euclidian distance formula. A Euclidian problem is symmetric because the distance is equal in both the direction the distance must be equal from City ‘A’ to City ‘B’ to City ‘B’ to City ‘A’. Therefore I must say that Symmetric Euclidian problem is undirected graphs.

If a problem is asymmetric then the distance is not equal from City ‘A’ to City ‘B’ to City ‘B’ to City ‘A’. Asymmetric problems are directed graphs there is also a weights presents on the edges of the graph.

There are some Problem where we unable to calculate the distance through Euclidian distance formula are called Random distance matrix problems.

Most researchers used the problems based on Symmetric Euclidian. I have assumed in this thesis that solve the problems given in the 2-D plane due to easy to write code for these types of problems and instances of TSP problems are easily available.

2.3 P-Class

P is the complexity class which contains problems that can be solved in polynomial time on a single deterministic Turing machine. Example of Problems is sorting.

2.4 NP-Class

The NP abbreviation refers to "non-deterministic Turing machine" .NP is the set of all decision problem for which the 'yes' answers have simple proofs to the fact that the answer is indeed 'yes'. These proofs have verifiable in polynomial time by a deterministic Turing machine. So from this we can say that the NP is the set of decision problems which can be solved in polynomial time by a non-deterministic Turing machine.

The complexity class P is contained in NP Class but NP contains many important problems.NP class also contains hardest of which are called NP-Complete problems. NP Complete problems
are those problems which have no algorithms that can solved problem in a polynomial time.

This implies that if a problem is NP-Complete there is no polynomial time algorithm for it unless P=NP. But this has not been proved yet, and it looks like that probably P not equal NP.
If it was possible to find a polynomial time algorithm for an NP-Complete problem, then All NP-Complete problems solved in polynomial time and P would equal NP. But still question is not proven yet. There is no existence of algorithm for NP-Complete problems. That means if TSP is NP-Complete there is no known polynomial time algorithm for it, in fact it is O (N!) where *N* is the number of cities.

2.5 Proof that Travelling Salesman Problem Is NP-Complete

In this thesis I am totally concentrate over on the TSP problems of Euclidean symmetric. C.H.Papadimitriou[1] proof that the Euclidian symmetric problems belongs to the class of the NP-Complete. He proof that due to the exponential factor O (N!) it is not possible to get accurate solution of TSP problems where 'N' is representing the total number of cities. It will take enough time to calculate solution of just 40 cities tour with in around $8.159*10^{47}$ operations.

2.6 Problem That Cannot Be Solved In Polynomial Time

It is very difficult to solve the TSP problem accurately. we can get the solution of a TSP problem in two way one is try to focus on small instances problem and the other way is that try to find an approximate solution of problems with in polynomial time. If we have concentrate over only on small instances problem then we can loss the chance to study some interesting problems by neglecting large instances problem because If we are trying to concentrate on large instances problem then we also learn and got a new ideas and techniques how to work on some practical problems.

2.7 Instances Used In This Thesis

The task of solving the TSP problem is not accurately feasible. I have experimented results in this paper on the problems that are based on symmetric Euclidian. I have collect Instances of such types of TSP problems from the following website[12] on which I have tested my experiment of the Multilevel Scheme to obtain a solution of Traveling Salesman Problem.

Chapter No.3

Heuristics Used For Solving Travelling Salesman

In this section I am trying to explain different types of solution that can be used to create solvers for traveling salesman problem. Main target is to present elements that are used later for constructing the Multilevel Scheme using these different strategies.

3.1 TSP Solvers

There are four different way of TSP solving methods

1. Tour Construction Method.
2. Local Search Method.
3. Backtracking-Jump Method.
4. Iterative Methods.

We will discuss some important methods related to each category.

3.2 Tour Construction Heuristics

A most simple method in which tour construction heuristic can create the tour by using an unordered collection of cities from the TSP Instance file. Although it can create the tour but does not try to improve the tour. Usually the output of the tour construction heuristic will be the input of the Local Search Method.

A tour construction heuristic can create the tour very efficiently but it's also depends on which construction heuristic you are using.

3.2.1 Nearest Neighbor

In this tour construction heuristic the salesman starts at some city and then visits the city nearest to the starting city and then he visits the nearest city that was not visited so far until all cities are visited and the salesman returns back to the starting point city.

It is probably close to the real salesman's approach but it is a poor heuristic because instead of selected cities which have a low cost distance they tried only to select the nearest neighbor

city whether the cost of selected city is low or high.

3.2.2 Random

In this tour construction heuristic the salesman select the first city randomly and after that make a tour by randomly select the another city and after the selection add this city to the tour and doing same thing until all the cities are now part of the tour. Random tour construction heuristic stops when all the cities are part of the tour.

3.2.3 Greedy

This tour construction heuristic works simply by growing a Hamiltonian cycle in the graph in which it is necessary that each vertex should be visited only one time.

Greedy can create this Hamiltonian cycle by first picking the shortest edge and then add this shortest edge to the tour until all the cities are included in the tour. Greedy can repeat this whole procedure until and less whole cities will be the part of the Hamiltonian cycle.

Greedy can create better tour as compared to the other heuristics Nearest Neighbor and Random.

3.3 Local Search Heuristics

A local search heuristic accept the output of tour construction heuristic as an input and using some strategy heuristic algorithm which will alter the tour and try to produce a tour shorter the previous one.

If they modified tour and if it is better than the given tour then it will continue to work on it else if not found any improvement then just forget it and tries to obtain a new modification of the solution.

There is no guarantee from any local search heuristics algorithm that they can return the

improvement in the input tour.

3.3.1 2-Opt

This tour construction heuristic systematically looks at each pair of non-adjacent edges of the current tour and determines whether the tour length would be decreased by removing these two edges and adding other two possible pair of edges which would be a better result in a tour. If so then exchange is performed and continues search for an improving pair of edges continues.

3.3.2 3-Opt

This is similar to the two-opt heuristic, but instead of looking and movement of two edges in the tour 3-Opt first chooses 3 edges from the tour and then remove these 3 edges from the tour and try to reconnect them in the new fashion as trying to create the shortest tour. It takes O (n^3) time to derive a new and better tour on the other hand 2-Opt takes O (n^2) time that's why 3-Opt heuristic is very slow as compared to the 2-Opt heuristic

3.4 Iterative Methods

There are some iterative methods which further improves the tour iteratively. In this section I have tried to explain two iterative methods Lin Kernighan Algorithm and Simulated Annealing Algorithm.

3.4.1 Lin Kernighan

Lin Kernighan is one of the best algorithms of TSP which will generate an optimal solution for the TSP. But the implementation of LK is not an easy task. There are various implementation technique through which LK can work and each technique also affect on the performance. The creation of LK algorithm is due to that static value of **"k"** in the K-Opt method does not provide the best solution because no one can know which value of **"k"** can gives us the best tour. Lin and Kernighan solve this problem by introducing the algorithm in

which value of the “k” variable is not fixed and the algorithm will change the value itself and used the value of “k” which will produce the better result. Lin Kernighan can change the value of **“k”** at run time.

This is the most powerful heuristic. It works as follows It removes an edge from the current tour, and try to add edges with the other node in such a way that will create the new shortest tour. By alternating an edges between nodes from the tour will create a new shortest path. These add/remove operation is then repeated. Let the "gain sum" be the sum of the lengths of the removed edges (except the last) minus the sum of the lengths of the added edges. This add/remove operation is continues until the sum is positive and when there are some remaining unadded/unremoved edges. After joining its endpoints the cost of tour is calculated for each construction and if it is less than that of the original tour then keeps track of this improved solution of the TSP. After the sequence of possible interchanges if an improvement was found at any point, then replaces the original tour by the best tour and the procedure is repeated.

3.4.2 Simulated Annealing

The term simulated annealing is also another technique which will be useful in refinement step of multilevel solution of TSP. This approach is derived by Monte Carlo it is used for minimizing functions with a large number of variables. The term Simulated annealing has comes from by analogous chemical process of heating and after that slowly cooling a substance this whole process created a crystalline structure. In simulation if the cost function is minimized then this corresponds to when the substance have crystallized.

Simulated Annealing decreases the temperature in stages until the system freeze and no improvement can be possible. In each temperature the simulation must take long enough time in reaching to a steady state this is known as thermal process. The temperature of the system will be decrease and the number of iterations applied to thermalize the system in each level of the temperature comprises an annealing scheduling. In order to implement Simulated Annealing it needs to first initialize with a specific configuration. New configuration is

constructed by taken the system into a new state. If the energy of this new state is lower than the previous energy level then this change is accepted unconditionally and the system is updates. But if the energy is greater than previous one then new configuration has a chance to be accepted it is known as Metropolis step and this is the fundamental procedure of Simulated Annealing. Through metropolis step we can allow the system to find area with lower energy states.

In 1983 computer scientist from IBM was first proposed that simulated annealing can also used for solving optimization problems. They suggested that simulated annealing is used for finding solutions to arbitrary optimization problems. Today Simulated Annealing algorithm is used in solving variety of problems. Simulated Annealing also solved the problem of TSP in an optimum time.

Johnson and McGeoch[2] have explained that simulated annealing is not so good in his implementation as compared to Lin Kernighan algorithm but solves the problems in good average time. They also told us that currently Simulated Annealing algorithm for TSP is not an optimal solution but it will be produce the optimal solution if we can change some different values of constant variables.

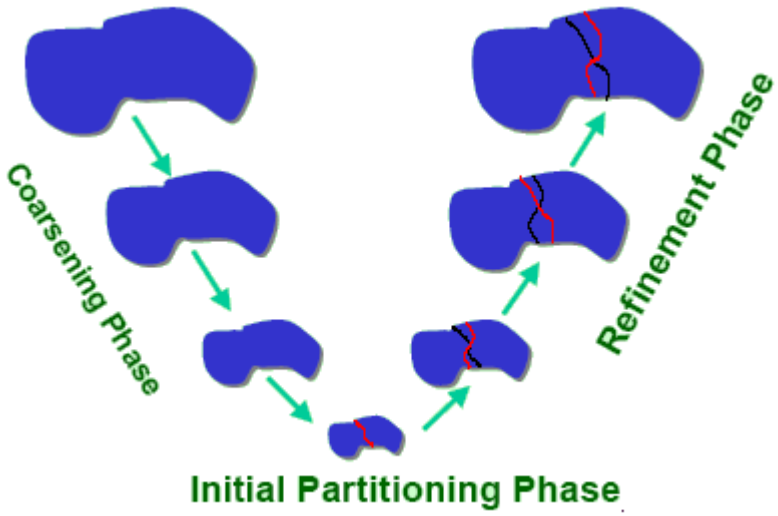
Chapter No.4

Multilevel Graph Partitioning Approach

In this chapter of the report I am presenting the multilevel scheme and also describe how we can adapt it to solve the Traveling Salesman problem. In this section I am trying to provide a full description to readers so they can easily understand about the multilevel scheme and also working on each step including coarsening, initialization, and refinement. At the end of this section of report we are focusing how we can solve the traveling salesman problem using the concepts of multilevel schemes.

4.1 What Is Multilevel Graph Partitioning??

In a Multilevel graph partitioning we are trying to reduce the size of the graph by collapsing vertices and edges means that to divide the graph into sub smaller graph and refine the partition during uncoarsen it to construct a partition for the original graph as shown in figure.



Firgure No.1

According to the figure during the coarsening phase we have first divided the graph into smaller partitions and coarsen it till when two vertices with a single edge remaining then we initialize the partition by adding an edge between these two vertices in the initial partitioning phase and after that we try to sum up these small graph solution to make a original graph. This produces a high quality solution after the top refinement phase.

As point to be noted that multilevel scheme is not a new strategy to solve the problem. It has been used successfully in the graph partitioning [4] and graph drawing [5] and others circuit solving problems.

4.2 The Multilevel Scheme Pseudo Code

Multilevel refinement (input problem instance P_0 ,0;output solution C_0 $\{P_0\}$)
Begin
l: = 1
While (coarsening)
P_{l+1} = coarsen (P_l)
l: =l+1
End
C_l $\{P_l\}$ = initialize (P_l)
While l>0
l: =l-1
C_l^0 $\{P_l\}$ = extend (C_{l+1} $\{P_{l+1}\}$, P_l)
C_l $\{P_l\}$ = refine (C_l^0 $\{P_l\}$, P_l)
end

4.2.1 The Coarsen Step

In the Coarsen step we decrease the size of the problem step by step and also the problem must not be trivialized.

4.2.2 The Initialize Step:

In the Initialize step we prepare the coarsened solution in such a way that it can be used in the

next step of the refinement solver.

4.2.3 The Refinement Step:

In the Refinement step we can apply the regular refinement algorithm on the solution of each step of uncoarsening.
The multilevel scheme (MLS) must also these two conditions.

Condition No 1:

Any solution in the coarsened stage should be a legitimate solution on the problem type. Means that it produces a legitimate solution if its only initialized and then extended without refinement to full size the cost of traveling must be equal to the original graph cost.

Condition No 2:

The coarsened problem must reflect the original problem through this it takes no more space.

Chapter No.5

How TO Solve The Traveling Salesman Problem

In order to solve the traveling sales man problem by using the idea that has given by multilevel scheme to divide the problem into multiple sub partitions and try to work it on individually Sub-Partitions. I have used the K-mean partitioning algorithm to try to divide the problem into multiple sub-partitions and after that solved each sub-partitions separately and finally used the information obtained by using solution of sub-partitions to improve the tour.

5.1K-Mean Partitioning

This algorithm was developed by J.MacQueen in 1967 and then J.A. Hartigan and M.A. Wong in near 1975. K-means clustering is an algorithm which is used to classify or group objects based on the attributes / features into K number of partitions. K is positive integer number. It's simply work by minimizing the sum of squares of distance between data and the cluster centroid. The overall purpose of K-mean partitioning is to classify the data. K-mean is one of the simplest partitioning schemes.

5.2 Why Used K-Mean Partitioning

The main objective and theme of this method is to divide the graph into smaller partitions and based on the concept that it first divides the tsp instance problem into multiple sub-partitions by dividing the total number of the cities that contain tsp instances by the number of sub problem you want and assign the nodes to the partitions which is near to the specific cluster.

5.3 Coarsening Step By Using K-Mean Partitioning

The K-mean is a very popular method for partitioning through which I am dividing problems into multiple sub problems. In this section I am explaining the detail how the K-Mean Partitioning algorithm is working.

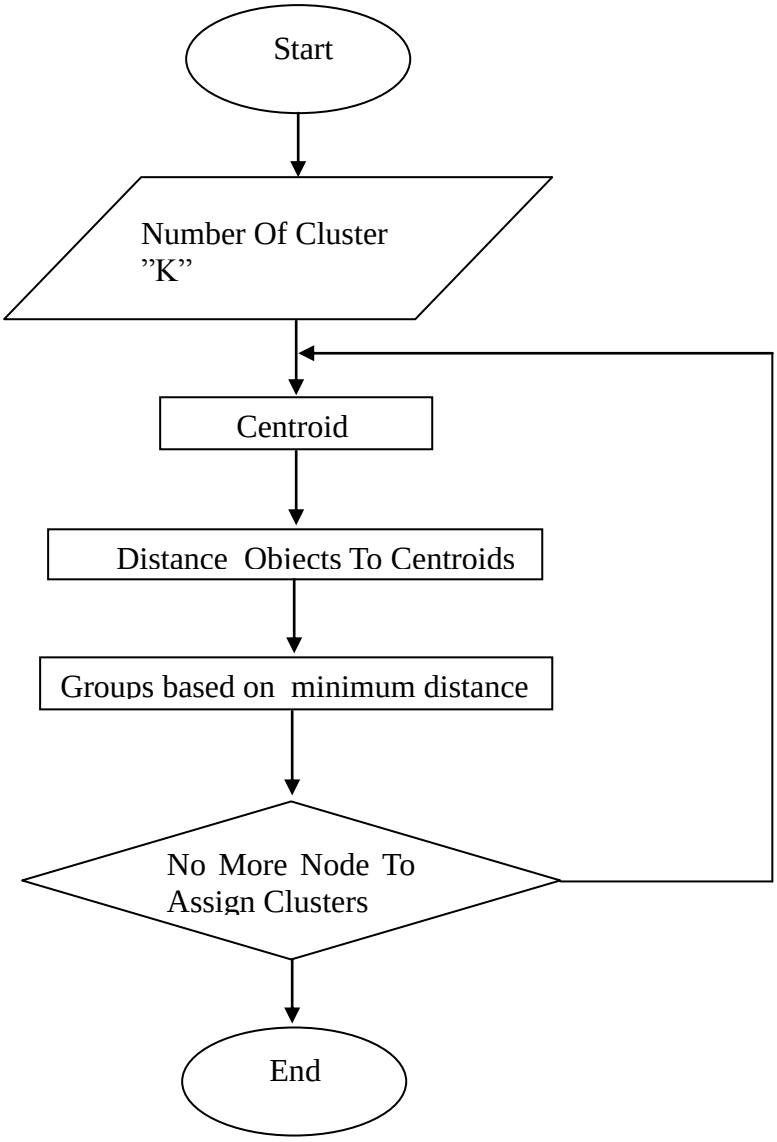


Figure No.2

The above mentioned flow chart diagram shows that how the k-mean partitioning algorithm divides the main tsp instance problem of the traveling salesman problem into multiple sub-problems. The K-mean partitioning algorithm can start work by obtaining a value of k which can be calculated by dividing the total number of cities the tsp instance have by 2 to trying to divide it into maximum partitions. First assign all the nodes to the guess cluster and after that

randomly select some nodes as the center point of the partition and assign nodes to the partitions which are near to the specific partition and assign all the nodes to the cluster in this manner.

5.3.1 How To Calculate The Total Number Of Partitions

It is calculated by using the following formula
Number of Partitions= Total No. Of Cities in Tsp Instance / 2.

Through this I have tried to divide the tsp instance problem into maximum number of sub-problem.

5.3.2 Working Of How To Assign Nodes To Each Sub-Partitions

Due to the large number of the cities tsp instance problem can contain so I am considering these 4 four cities to explain that k-mean partitioning problem are working.

Let us suppose that we have four instances of traveling salesman problem having following coordinates as shown in Figure No.2

City1 = (1, 1)
City2= (2, 1)
City3= (4, 3)
City4= (5, 4)

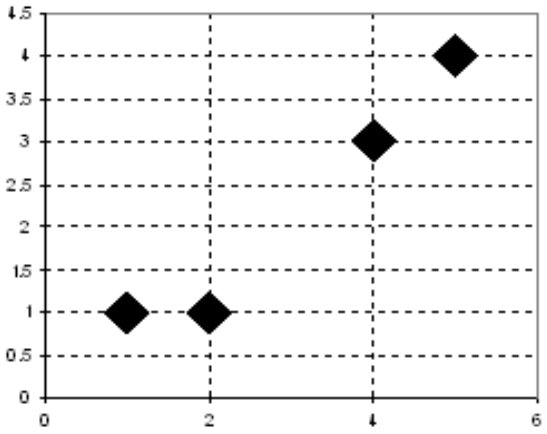


Figure No.3

So we have four instances so it will be divided into 2 partitions.

5.3.3 Iteration No. 0 :How To Select Centroid

The first step is randomly select two cities as the centroid of the problem. Let us suppose that we are using city1 and city2 as the centroid of the problem as shown in Figure No.3

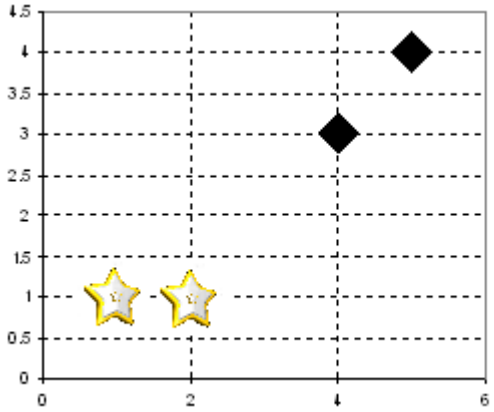


Figure No.4

5.3.3.1 Iteration No. 0:How To Solve The Distance Between Centroid And The Cities

Now after the selection of the centroid in first iteration number 0 now we need to calculate the Euclidean distance between centroid and cities. Calculate the distance in this way.

5.3.3.1.1 Distance Between Centroid1 And Cities

City1 with centroid 1 = 0

City2 with centroid 1= $Sqrt((2-1)^2+(1-1)^2)=1$

City3 with centroid 1 =$Sqrt((4-1)^2+(3-1)^2)=3.61$

City4 with centroid 1 =$Sqrt((5-1)^2+(4-1)^2)=5$

5.3.3.1.2 Distance Between Centroid2 And Cities

Distance calculated with the 2nd centroid in a similar way

City1 with centroid 2 =Sqrt($(1-2)^2+(1-1)^2$)=1

City2 with centroid 2 =0

City3 with centroid 2 =Sqrt($(4-2)^2+(3-1)^2$)=2.828

City4 with centroid 2 =Sqrt($(5-2)^2+(4-1)^2$)=4.2426

5.3.3.2 Iteration No. 0:The Distance Matrix Representing Distance Between Centroid And Cities

So the distance matrix we have

$$D^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.828 & 4.24 \end{bmatrix}$$

Representing the distance for each city from centroid1
Representing the distance for each city from centroid2

The first row of the distance matrix is representing the cost of the distance between cities and the first centroid and the 2nd row is now representing the cost of the distance between cities and second centroid.

5.3.3.3 Iteration No. 0:Grouping Of Cities

As it is clearly shown in the distance matrix that distance of city3 and city4 to the centroid2 is quite minimum as compared to the centroid1. As according to the rule of k-mean partitioning assign each city based on the minimum distance with the centroid. So assign city1 to the first group, city2 to the second group, assign city3 to the second group and also assign city4 to the 4th group as shown in figure 4.

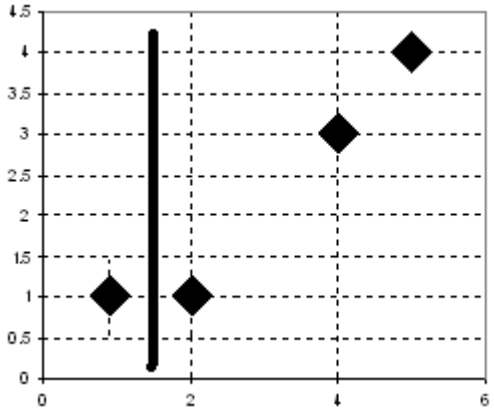


Figure No.5

So the group matrix after this step is

$$G^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

City1 City2 City3 City4

First row of the group matrix is representing how many cities they have combine and the second row of the matix is representing how many cities group 2 have.

5.3.4 Iteration No 1: How To Calculate The Centroid

After this on next iteration k-mean partitioning again calculates the centroid in each group to more classify the data. So there is only one city in group one so first centroid is against city number one due to have only single city and the other centroid is calculated by summing the x-coordinates and y-coordinates with the total number of the cities that the group have. So the 2nd centroid in the group 2 is calculated by summing the coordinates in the group 2 in this way C (2+4+5/3,1+3+4/3)= C(3.66, 2.66) .

5.3.4.1 Iteration No 1: Again Need To Calculate The Distance

There is once again need to calculate the distance between centroids and cities to further classify the problems.

5.3.4.1.1 Distance Between Centroid1 And Cities

City1 with centroid 1 = 0

City2 with centroid 1= Sqrt((2-1)^2+(1-1)^2)=1

City3 with centroid 1 =Sqrt((4-1)^2+(3-1)^2)=3.61

City4 with centroid 1 =Sqrt((5-1)^2+(4-1)^2)=5

5.3.4.1.2 Distance Between Centroid2 And Cities

City1 with centroid 2 =Sqrt((1-3.66)^2+(1-2.66)^2)=Sqrt(7.0756+2.7556)=3.14

City2 with centroid 2 =Sqrt((2-3.66)^2+(1-2.66)^2)= Sqrt(2.7556+2.7556)=2.36

City3 with centroid 2 =Sqrt((4-3.66)^2+(3-2.66)^2)=Sqrt(0.1156+0.1156)=0.47

City4 with centroid 2 =Sqrt((5-3.66)^2+(4-2.66)^2)=Sqrt(1.7956+1,7956)=1.8950

5.3.4.2 Iteration No 1: The Distance Matrix Representing Distance Between Centroid And Cities

This is showing the Distance Matrix at iteration number 1 this can be filled by calculating the distance between centroids and cities. So the distance matrix at this point is

$$D^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.890 \end{bmatrix}$$

5.3.4.3 Iteration No 1: The Grouping Matrix

So assign the cities to the group according to the minimum distance. Assign city 1 to the first centroid, second city to the first centroid, third city to the second centroid and fourth city to the second centroid as shown in the figure no.5.

$$G^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

City1 City2 City3 City4

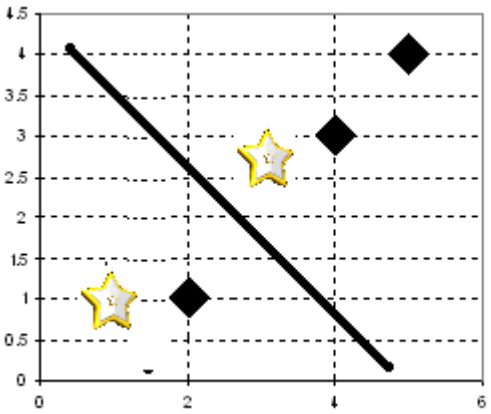


Figure No.6

5.3.5 At Iteration No.2 :Calculating The Centroid Once Again

So there is still need to classify data more because grouping of cities at step 1 and step 2 is not same so we are again calculating the centroid for each group again by summing the coordinates of the cities that group1 and group 2 have and divide it by total number of the cities.
Group 1= centroid 1 = C(1+2/2,1+1/2) = C(1.5,1)
Group 2=centroid 2 = C(4+5/2,3+4/2) = C(4.5,3.5)

Now again need to calculate the distance from each city with the centroid 1 and centroid 2.

5.3.5.1 Iteration No 2: Again Need To Calculate The Distance

There is once again need to calculate the distance between centroids and cities to further classify the problems.

5.3.5.1.1 Distance Between Centroid1 And Cities

City1 with centroid 1 = $Sqrt((1-1.5)^2+(1-1)^2) = Sqrt(0.25+0)=0.5$

City2 with centroid 1= $Sqrt((2-1.5)^2+(1-1)^2)=Sqrt(0,25+0)=0.5$

City3 with centroid 1 =Sqrt((4-1.5)^2+(3-1)^2)=Sqrt(6.25+4)=3.20

City4 with centroid 1 =Sqrt((5-1.5)^2+(4-1)^2)=Sqrt(12.25+9)=4.61

5.3.5.1.2 Distance Between Centroid2 And Cities

City1 with centroid 2 = Sqrt((1-4.5)^2+(1-3.5)^2) =Sqrt(12.25+6.25)=4.30

City2 with centroid 2= Sqrt((2-4.5)^2+(1-3.5)^2)=Sqrt(6.25+6.25)=3.54

City3 with centroid 2 =Sqrt((4-4.5)^2+(3-3.5)^2)=Sqrt(0.25+0,25)=0.70

City4 with centroid 2 =Sqrt((5-4.5)^2+(4-3.5)^2)=Sqrt(0.25+0.25)=0.70

5.3.5.2 The Distance Matrix Representing Distance Between Centroid And Cities

Now the distance matrix contains following values after calculating the distance in iteration number 2 between centroids and other cities.

$$D^2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.7 & 0.7 \end{bmatrix}$$

5.3.5.3 The Group Matrix

Now again assigning the cities to the groups according to the minimum Euclidean distance.

$$G^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

City1 City2 City3 City4

5.3.6 When K-Mean Partitioning Stop Partitioning

Thus again cities riches to the same group as they did in the previous iterations city1 and city2 again reaches to the same group1 and city3 and city4 again in the group2.

Mean that $G^1=G^2$ So there is no need to grouping again thus k-mean partitioning reaches their ending step and there is no need to move any city to the other group. Therefore K-Mean partitioning has reached their stability and no more iteration is required. So the finally results produced by K-Mean Partitioning is shown in the table.

Name Of the City	Group Number
City1	Group 1
City2	Group 1
City3	Group 2
City4	Group2

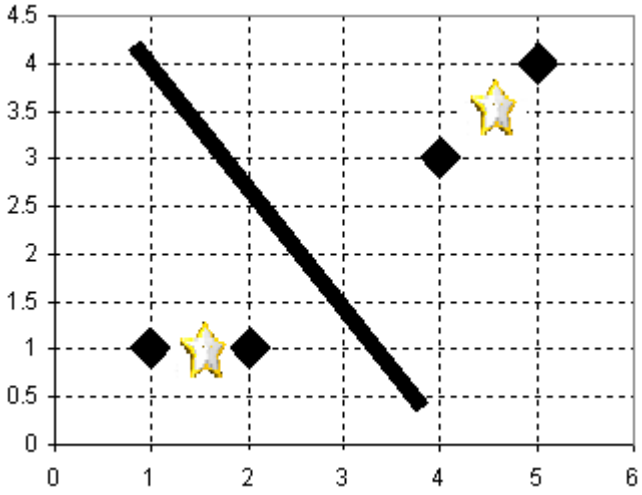


Figure No.7

This figure shows that partitioning has done by the k-mean partitioning algorithm.

In that way K-mean Partitioning has created partition by using the tsp instance problems.

5.4 Uncoarsening Phase

The basic purpose of the Uncoarsening part of the multilevel scheme is that to solved each partitions separately and apply Lin Kernighan algorithm on the information obtained by using the solve partitions .

5.4.1 Solved Each Sub-Partitions

After the separation of the problems into multiple sub partitions there is need to solve each partition separately.

I have used the Greedy tour construction heuristic to take a good solution of individual small Partitions because Greedy is the best tour construction heuristic. Also I have used the Lin Kernighan algorithm to further improve the total length of tour. As shown in the Figure No.7 that each partition is now solved separately.

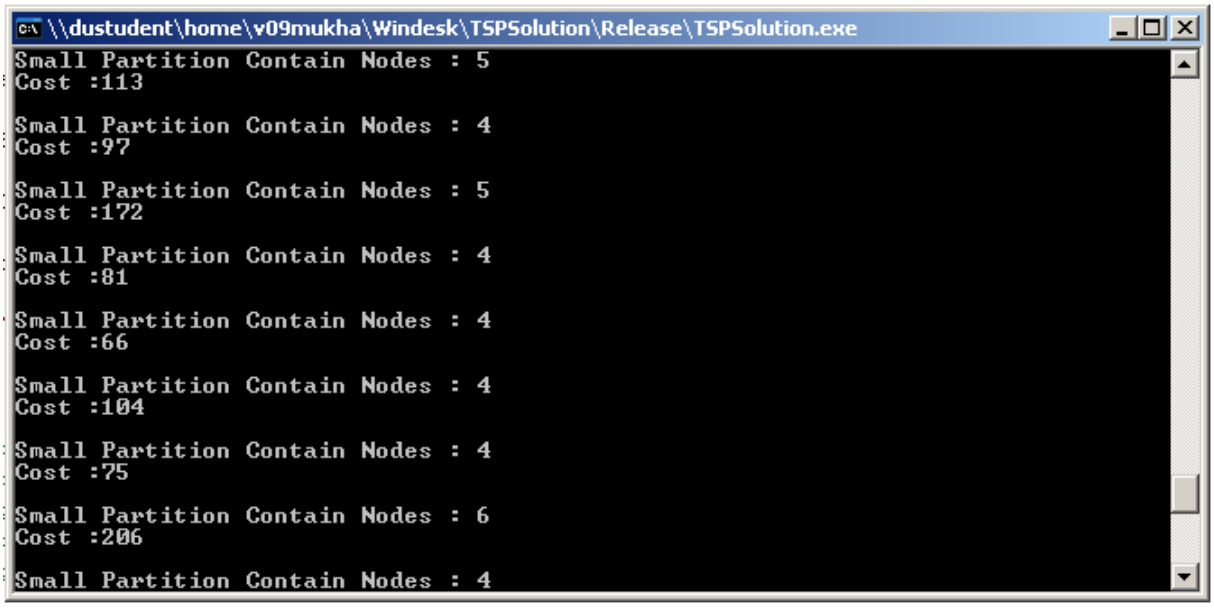


Figure No.8

See in the figure that each sub-partition is now solved independently.

After solving each partition step by step I have simply recreating the graph once again by simply add each solved partition back to the graph..

At the end I have applied the Lin Kernighan algorithm on the information obtained by using solved partitions.

5.4.2 Lin Kernighan Algorithm

I have used the Lin Kernighan algorithm to further improve the tour that has been generated by using the information obtained by applying k-mean partitioning algorithm.

5.5 Why Used Lin Kernighan Algorithm

In experiment of this thesis I have seen a lot of differences in the results given by the iterative methods. Before using the Lin Kernighan algorithm I had used the Simulated Annealing Algorithm but through this I had not got results closed to the optimal solution that's why I have used the Lin Kernighan algorithm.

Lin Kernighan is one of the best heuristics which solves the Euclidean traveling salesman problem and nearly give us the optimal solution. It can create a new tour by swapping pairs of sub tours. Lin Kernighan uses the k-Opt local search heuristic as already defined that they are simply working by switching paths and try to make a new shorter tour.

Lin Kernighan decides at each step that how many between cities need to be switched to find a better and shorter tour.

5.6 Euclidean Distance Formula

I have used the following Euclidean formula in this thesis to calculate the distance between clusters and cities.

$A = (C_x - D_x)^2$

$B = (C_y - D_y)^2$

Square Root (A+B)

Chapter No.6

Experimental Results

In this part of the report I am presenting the results which is obtained by the using the solutions obtained by using the k-mean partitioning approach. In experiment I have compare the results between division of problem into maximum number of sub-problems and division of problem into limited number of sub-problems. Also compares the results obtained by using different 3-opt and 2-opt local search heuristics in the refinement step of the multilevel graph partition.

6.1 System Information

All test runs were done on a PC Pentium Intel(R) Core(TM) 2 Quad with a 2.66 GHz processor and 1.97 GB of Ram, using Windows XP operating system and the Microsoft Visual Studio. I have used the source code for testing of experiment of k-mean partitioning which is freely available on the website [13] .I have adapted it with some changes in the code according to the requirement of the thesis. I have collect Instances of such types of TSP problems from the following website [13] on which I have tested my experiment.

6.2 Comparison Of Result By Dividing It into Minimum and Maximum Partition

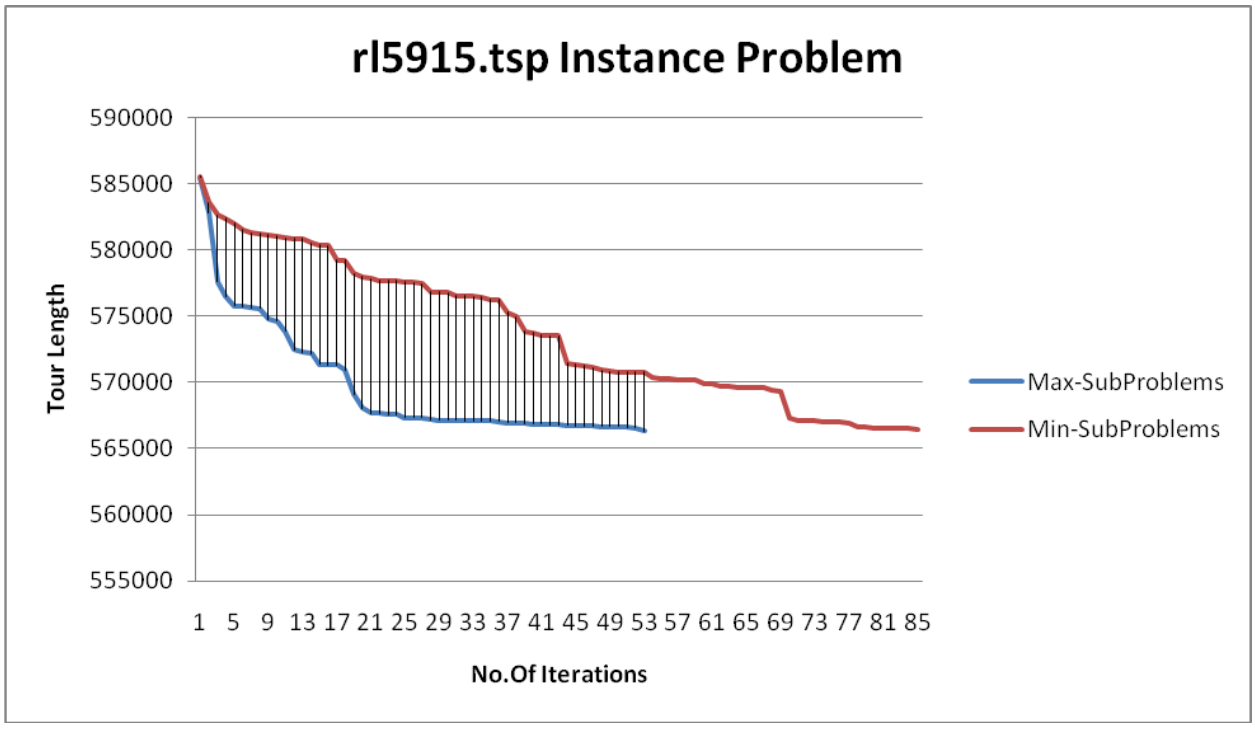


Figure No.9

Above chart shows the comparison between to divide the tsp instance problem into maximum no of sub-problems and minimum number of sub-problem by using the k-mean partitioning algorithm. The red line shows that "rl5915.tsp" instance problem is divided into minimum number of sub-problems and blue line indicates that problem is divided into maximum number of sub-problems. Through this comparison chart I am trying to proof that through division of problem into maximum number of sub-problem by using k-mean partitioning can give us a better and optimal tour length which is 566300Km but also solves the problem very efficiently and faster as compared to the problem which is divided into minimum number of sub-problems. The problem which is divided into minimum number of sub-problem can take a lot of time to generate the solution. It also gives the good solution which is 566451Km but not good enough comparatively result obtained by dividing problem into maximum number of

sub-problem. Therefore it proof's that if we divide the problem into maximum number of sub-problem it will give us an optimal results within a very few times so its proofs that k-mean graph partitioning approach is really working well and it's very suitable solution for the problem of the traveling salesman problem.

6.3 Experiments By Applying Lin Kernighan Algorithm On The Information Obtained By Graph Partitioning Scheme

In this section of the report use the information obtained by solving the sub problems and applying the Lin Kernighan algorithm I got these got these results.

S.No	TSP Instance	OutPut	Optimal Length
1	Berlin52.tsp	7542	7542
2	rl1304.tsp	253010	252948
3	fl1400.tsp	20164	20127
4	u1432.tsp	153159	152970
5	Pcb3038.tsp	137768	137694
6	d1655.tsp	62128	62128
7	vm1748.tsp	336941	336556
8	dl5112.tsp	1575596	1573084
9	vm1084.tsp	239396	239297
10	u2152.tsp	64355	64253
11	usa13509.tsp	20018711	19982859
12	u2319.tsp	234683	234256
13	pr2392.tsp	378107	378032
14	u1817.tsp	57331	57201
15	rl5915.tsp	566962	565530
16	u1060.tsp	224121	224094
17	rl1889.tsp	316549	316536
18	rl1323.tsp	270646	270199
19	Pr1002.tsp	259045	259045
20	pla7397.tsp	23269614	23260728
21	Pcb1173.tsp	56892	56892
22	nrw1379.tsp	56662	56638

23	fl1577.tsp	22256	22249
24	fl3795.tsp	28853	28772
25	d2103.tsp	80513	80450
26	d1291.tsp	50864	50801
27	fnl4461.tsp	183040	182566
28	dsj1000.tsp	18660188	18659688
29	rl5934.tsp	556678	556045

Above results I have got are clearly shows that I have got a very good result by the experiment of the multilevel graph partitioning approach. I have got an optimal solution of the instance problems **berlin52.tsp, d1655.tsp, pcb1173.tsp** and also got solution nearly equal to the optimal solution of the tsp instance problems having names **d1291.tsp, fnl3795.tsp, d2103.tsp, fl1577.tsp, nrw1379.tsp, rl1323.tsp, rl1889.tsp, u1060.tsp, u1817.tsp, pr2392.tsp, u2152.tsp, vm1084.tsp, vm1748.tsp, fl1400.tsp.**

Also got the good solutions of the large collection of cities of the tsp instance problem having names **fnl4461.tsp, rl5934.tsp, pla7397.tsp, rl5915.tsp, usa13509.tsp, d15112.tsp, rl1304.tsp** instances.

Therefore above results proof that k-mean partitioning is really good for traveling salesman problem and through this I got the good optimal results.

Chapter No.7

Overview On Experiment

In this chapter I am trying to explain some factors which play a major role in the creation of the results. Firstly division of problems into multiple sub-problems by using a large scale data clustering based k-mean algorithm, solved each sub-problem by using greedy tour construction heuristic, and apply Lin Kernighan algorithm on the information's obtained by using Lin Kernighan algorithm on it..

7.1 Division Of Problems Into Multiple Sub-Problems

As from the result I get during experiment it proofs that it is very beneficial for us to divide the problem as soon as possible into multiple sub-problems through which it is possible to solves the traveling salesman problem within a few second and also get an optimal solution as compare to the solution which is obtained by dividing it into limited number of sub-problems. Thus it's proof that through multilevel scheme we can solve this intractable problem within a few minutes.

7.2 Adaption Of Lin Kernighan Heuristic

Every One knows that Lin Kernighan is a well known and best heuristic for solving the traveling salesman problem. Before using this heuristic I had also tested Simulated Annealing Algorithm in the refinement step of the multi-level scheme. But I was faced a lot of problem by using Simulated Annealing Algorithm. There is a lot of problem to adjust their adjustable variables just like temperature decrement factor, search length and to set the limit of the temperature. First we need to set these parameters which need a lot of experiment to set the values of these variables.

After all the setting I can't receive the good optimal results as Simulated Annealing is not a good algorithm which produces the good solution for the traveling salesman problem as compared to the Lin Kernighan algorithm. I have learned a lot by working on the solution of the traveling salesman problem using multilevel graph partitioning scheme. There is no doubt Lin Kernighan is the best heuristic which gives me just amazing results. Only due to the Lin Kernighan Algorithm I got very good solutions.

7.3 Conclusion

In this thesis I have tested multilevel scheme for the solution of traveling salesman problem. I have first divide the problem into multiple sub-problems by using the algorithm of the k-mean partitioning and further solves the each sub-problem by using the Greedy tour construction heuristic and then finally applies the Lin Kernighan algorithm in the refinement step on the information that can be got by solving each individual small partitions. I got the optimal solutions of some tsp instance problems and nearly got the optimal solution of various problems. Also I got a good solution of the big tsp instance problem. K-mean creates a very good partition and Greedy which is the best tour construction can solves each sub-problems and Lin Kernighan algorithm which is the best heuristic can further improves the tours by using the information given by k-mean partitioning. Over all we can say that it is possible to solve the traveling salesman problem through graph partitioning scheme and to solves this intractable problem into a few minutes by using this strategy.

References

[1] Euclidean TSP is NP-complete.

[1] Theoretical Computer Science, 4:237-244,C. H. Papadimitriou 1977

[2] The Travelling Salesman Problem: A case study in local optimization by

David S. Johnson and Lyle A. McGeoch 1995

[3] METIS_A Software Package for Partitioning UnstructuredGraphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices Version 4.0

[4] An experimental comparison of Different graph coarsening schemes

Noureddine Bouhmala

[5] A Multilevel approach to the travelling salesman problem

Mathematics research report 00/IM/63

Chris Walshaw 2000

[6]A Multilevel Lin-Kernighan-Helsgaun Algorithm for the

Travelling Salesman Problem

Chris Walshaw

Computing and Mathematical Sciences, University of Greenwich,

Old Royal Naval College, Greenwich, London, SE10 9LS, UK.

[7] Web Source:

http://www.math.princeton.edu/tsp/concorde.html

William J. Cook (bico@math.princeton.edu) (Last Update 3 July 2001.)

[8] A Multilevel Approach to the Travelling Salesman Problem

ChrisWalshaw *Computing and Mathematical Sciences, University of Greenwich,*

Park Row, Greenwich, London, SE10 9LS, UK.

[9] Correspondence: Simulated Annealing

Noureddine Bouhmala 2001

[10] Web Source:

http://www.math.princeton.edu/tsp/concorde.html

William J. Cook (bico@math.princeton.edu) (Last Update 3 July 2001.)

[11] Web Source:

http://www.research.att.com/~dsj/chtsp/download.html

Anonymous (Last Update 28 May 2002)

[12] RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

[13] http://www.akira.ruc.dk/~keld/research/LKH/

TSP INSTANCE PROBLEM

Below is the chart of the TSP Instance problems which shows the total number of cities and optimal length of each TSP Instance problem.

S.No	TSP Instance	Total Number Of Cities	Optimal Length
1	Berlin52.tsp	52	7542
2	rl1304.tsp	1304	252948
3	fl1400.tsp	1400	20127
4	u1432.tsp	1432	152970
5	Pcb3038.tsp	3038	137694
6	d1655.tsp	1655	62128
7	vm1748.tsp	1748	336556
8	dl5112.tsp	15112	1573084
9	vm1084.tsp	1084	239297
10	u2152.tsp	2152	64253
11	usa13509.tsp	13509	19982859
12	u2319.tsp	2319	234256
13	pr2392.tsp	2392	378032
15	u1817.tsp	1817	57201
16	rl5915.tsp	5915	567791
17	u1060.tsp	1060	224094
18	rl1889.tsp	1889	316536
19	rl1323.tsp	1323	270199
20	Pr1002.tsp	1002	259045
21	pla7397.tsp	7397	23260728
22	Pcb1173.tsp	1173	56892
23	nrw1379.tsp	1379	56638
24	fl1577.tsp	1577	22249
25	fl3795.tsp	3795	28772
26	d2103.tsp	2103	80450

27	d1291.tsp	1291	50801
28	fnl4461.tsp	4461	182566
28	dsj1000.tsp	1000	18659688
29	rl5934.tsp	5934	556045