ABSTRACT

BANSAL, AMOGH. Design of a Public Logistics Network. (Under the direction of Dr. Michael G. Kay.)

This thesis presents a design for a public logistics network (PLN) covering the continental United States. A systematic approach is developed to determine the initial national road network using only the Interstate highways and part of the U.S. highways. Various heuristics are developed for generating the underlying road network. In two of the heuristics, every Interstate node is used and then U.S. highway nodes are added to the road network to supplement the Interstate nodes. Another heuristic generates the network by directly joining the roads (based on shortest time routes) between the cities of more than certain population. The results from testing show that the later heuristic performs better than the former ones. This road network is then used for developing the PLN by selecting some of its nodes as the locations for distribution centers (DCs). The PLN is then developed by removing and adding arcs in a reduced network obtained by Delaunay triangulation of the selected DC nodes in the underlying road network. Given the number and location of DCs on this underlying network, the minimum average transport time for a package is used as the criterion to compare alternative PLN designs. The package demand used to determine the minimum average transport time is proportional to the population at each five-digit zip code centroid surrounding each DC. Effects of different parameters on the design of the PLN are studied. Finally, a genetic algorithm is used to get the optimal public logistics network for the entire U.S.

Design of a Public Logistics Network

BY

AMOGH BANSAL

A thesis submitted to the graduate faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Master of Science

OPERATIONS RESEARCH

Raleigh, NC
May 2004

APPROVED BY

Dr. Russell E. King	Dr. Negash Medhin

Dr. Michael G. Kay
(Chair of Advisory Committee)

BIOGRAPHY

Amogh Bansal, born on the 31st of May 1981 in the city of Deeg in the state of Rajasthan, India, completed his schooling from the city of Jaipur, India. He obtained his Bachelor of Technology degree in Mechanical Engineering from IIT Bombay.

During his graduate studies at North Carolina State University, he undertook course work in Operations Research and Industrial Engineering and worked in the area of developing meta-heuristics for logistics network design problems under the direction of Dr. Michael Kay. During his graduate studies he did an internship at Solectron Corporation in Creedmoor, NC. His research interests include logistics network design, discrete optimization and discrete event simulation.

ACKNOWLEDGEMENTS

I would like to thank the following people for their generous contributions to the completion of this work

- My parents Dr. Madan Mohan Bansal, Ms. Sudha Bansal and my brother Gaurav for your love and support.

- Dr. Michael Kay as my committee chair and advisor, you have provided unparalleled guidance and support during the course of the project. It has surely been a long road, but I have truly enjoyed and gained much from the experience.

- My committee members Dr Russell King and Dr. Negash Medhin for their encouragement and feedback.

I would also like to mention my friends Arvind, Jayesh, Bhasker, Vijayendra, Ujjwal and Shrikanth for their valuable company, which I shall cherish forever.

TABLE OF CONTENTS

LIST OF FIGURES vi

LIST OF TABLES viii

1. Introduction 1

1.1 Introduction 1
1.2 Objectives 2
1.3 Background 2
1.4 Approach 4

2. Design of Underlying Road Network 5

2.1 Heuristics to Design Underlying Road Network 7
2.1.1 Method I 8
2.1.2 Method II 10
2.2 Addition to Above Heuristic 11
2.3 Removing Unconnected Nodes 12
2.4 Method III 12

3. Developing PLN and Estimating Average Travel Time. 15

3.1 Developing PLN 15
3.2 Estimating Average Travel Time 22

4. Parameter Selection and their effects on Quality of Solution 27

4.1 List of Parameters 27
4.2 Checking the quality of Networks 32

4.3 Estimating the proximity factor 35

5. GA Experiments and Analysis 36

5.1 Genetic Algorithm 36
5.2 Improving the GA performance 39
5.3 Results 39

6. Conclusion and Future Work 44

6.1 Conclusion 44
6.2 Future Work 44

BIBLIOGRAPHY 46

APPENDIX 1A 47

APPENDIX 1B 70

LIST OF FIGURES

Fig 2.1 Interstate highways in U.S. 5

Fig 2.2 U.S. highways 6

Fig 2.3 Part of Interstate Network with *Cnode* at the center and the area chosen around it to add U.S. highways to this network. 9

Fig 2.4 Interstates and U.S. highways Road Network obtained by method II 10

Fig 2.5 Underlying Road Network when parameter value of (D_2 - D_1) is 140 mi 11

Fig 2.6 Underlying road network with joining cities of population 20,000 or more 13

Fig 3.1 Delaunay triangulation of U.S. Cities with population 100K or more 17

Fig 3.2 Removal of Arcs from the network—dashed lines show the removed arcs from the network. 18

Fig 3.3 Addition of arcs in the network—black double dotted dashed lines show the new additions to the network. 19

Fig 3.4 Flowchart shows steps involved in getting a public logistics network that will be evaluated for average travel time. 20

Fig 3.5 MATLAB Pseudo-code for developing PLN 21

Fig 3.6 MATLAB Pseudo-code for estimating average travel time 26

Fig 4.1 Distance Parameter (D2-D1) vs. No. of nodes/arcs in the network 31

Fig 4.2 Sorted distances of the nodes to its closest node for the network in Fig 2.6 32

Fig 4.3 Sorted distance of 200 randomly chosen DCs to their closest DC in the network of Fig 2.6 33

Fig 4.4 Weighted distances of 200 randomly distributed DCs in network of Fig 2.6 34

Fig 5.1 273 DCs selected in public logistics network with avg. travel time of 9.07 hrs 41

Fig 5.2 PLN of 273 selected DCs 41

Fig 5.3 The best and the average of the Average Travel Time vs. Number of nodes in the network L/U time of 5 min. 42

Fig 5.4 The best and the average of the Average Travel Time vs. Number of nodes in the network L/U time of 10 min. 43

Fig A.1 GA simulation results when L/U time is 10min. 74

LIST OF TABLES

Table 3.1 Parameters value used in estimating average travel time. 25

Table 4.1 Parameters used in developing URN, PLN 28

Table 4.2 Assumptions on the speed of roads: 29

Table 4.3 Number of nodes and arcs in the network 30

Table 4.4 Number of nodes and arcs in network with different values of parameter (D_2 - D_1) 31

Table 5.1 Results of GA run using Method II. 39

Table 5.2 Results of GA run with L/U time of 10 min. 40

Table 5.3 Results of GA run with L/U time of 5 min 40

Table A.1 Trace of the GA simulation..... 71

1. Introduction

1.1 Introduction

A public logistics network (PLN) is proposed for fast, flexible, and low-cost means of parcel transport. It consists of multi-company distribution centers and can help in alleviating traffic congestion and saving energy and labor costs. These facilities allow more efficient logistics systems to be established and they facilitate the implementation of advanced information systems and cooperative freight systems.

The distribution centers (DCs) in a public logistics network would be constructed in the vicinity of highways interchanges and throughout metropolitan areas and would help in reducing the number of vehicles transporting small loads of goods. By consolidating terminals, the implementation of advanced information systems and especially the cooperative operation of freight transportation systems is more practical. Similar to the dynamic pricing used to sell airline seats, a price for each available space on a truck and storage space at a distribution center could be negotiated in real time for each individual item. A PLN could be used by third-party logistics providers or companies that have established cooperative contracts. It also helps small and medium size enterprises to implement efficient freight transportation through the mechanization and automation of materials handling. These are systems in which a number of shippers or freight carriers jointly operate freight vehicles or freight terminals or information systems to reduce their costs for collecting and delivering goods and provide higher level of services to their customers. A PLN is likely to be most suitable for managing the multitude of commodity-like items (replacement parts, etc.).

1.2 Objectives

The main objectives of this research are:

1. To determine an initial national road network, where the road intersections are the candidate sites at which to locate the DCs used in the PLN.

2. Determine the number and location of DCs that result in the minimum average transport time for packages using the resulting PLN.

The research presented in this thesis extends the 36-DC network developed in [5] (covering only the southeastern part of the country) to cover the entire U.S. The 36-DC network was developed to determine the advantages of PLN over other private networks presently used by UPS and FedEx.

1.3 Background

A public logistics network is proposed as a means to extend many of the features of public warehouses to the entire supply chain in [5]; a 36-DC hypothetical public logistics network comprising southeastern U.S. was developed. The average transport time of this network is compared to the hub-and-spoke and a point-to-point network covering the same region. It was found that the public logistics network provided the minimum average transport time when the time required for loading/unloading at each transshipment point in the network was short.

Barrodale Computing Services Ltd. (BCS) has developed a procedure for automated generation of a road network within a geographic region. This procedure uses a digital elevation model (DEM) and 3-D feature data (hydrography, breaklines) to determine realistic routes that interconnect a number of pre-specified points in the region. The initial application

of this procedure is in spatially explicit timber supply modeling, to support the need for a forest road network that connects the cut blocks [1]. Once the points to be connected have been specified, the automated roading procedure involves three stages: In the first stage, those pairs of specified points that are considered to be adjacent are identified by Delaunay triangulation of the points. Each edge in this triangulation defines an adjacency. In the second stage, the optimal (minimum-cost) path through the region between each pair of adjacent points is calculated. Dijkstra's shortest-path algorithm is then applied to all the pre-specified adjacent point pairs to determine the minimum-cost between each pair of adjacent points. In the third stage, the selection of which of these paths to use in the final network is made, resulting in a road tree that connects all the pre-specified points.

Genetic algorithms (GA) have been shown to be a promising approach for a wide range of search and optimization problems [4]. It has been successfully applied to wide range of applications including warehouse problems and location-allocation problems. A genetic search for an optimization problem starts with a randomly generated initial population in which members and subsequent generations are duplicated or eliminated according to their fitness values. Further generations are created by applying GA operators. Through limited times of evolutions, a generation is obtained that contains solutions to the optimization problem.

There are three main operators in genetic algorithms. The first is the reproduction operator, which makes one or more copies of any member possessing a high fitness value, and members with low fitness values are eliminated from the solution pool. The second operator is the crossover operator. It selects two members as parents within the generation and several crossover sites to perform a swapping operation, generating new members called off springs that possess some traits inherited from both parents. The third operator is the mutation operator. It is performed by toggling one or more bits of some member selected at random. In a GA search procedure, reproduction plays the role of keeping better solutions from one generation to the next and crossover accelerates the search, while mutation explores some of unvisited points in the search space. The applications of these operators to the solution population make GA have the following advantages over the general optimal algorithms. It

starts search not from a single point, but from a set of points, allowing for parallel and global search; it is not restricted by the property of objective functions like other algorithms; it uses a probabilistic transition rule to progress from one search space to the next; the speed of convergence is fast and the likelihood of finding the global optimal solutions is greatly enhanced, etc.

1.4 Approach

This thesis presents the designing of national public logistics network (PLN) using meta-heuristics and genetic algorithms (GA). A systematic approach is developed to determine the initial national network using only the Interstate highways and part of the U.S. highways. The criterion that is used to determine the initial design of the network is the maximum coverage of the entire nation with all the interstates and only a part of U.S. national highways and to keep an upper bound on the size of the entire network used. The next motive is to determine the number and location of distribution centers (DCs) on this underlying network that results in the minimum average transport time. A genetic algorithm is used for this purpose with MATLAB as the tool for simulation.

Various heuristics are developed for generating the underlying road network. In the method I and II, every interstate node is tested for adding the U.S. highways in the road network. The method III generates the network by directly joining the roads (shortest time routes) between the cities of more than certain population. The results from the genetic algorithms show that the method III performs better than method I and II. The future scope and enhancements possible for these techniques to obtain better results are also discussed.

2. Design of Underlying Road Network

For design of a public logistics network (PLN), an underlying Road Network is needed on which distribution centers (DCs) can be placed. So, it is decided to include all the interstate highways in the underlying road network (URN), as this is the major road network with fast moving traffic on it. However, using only interstates highways does not provide enough roads for designing the underlying network as can be seen in the Fig 2.1. The Interstate road network is dense in the eastern part of the country and is sparse in the central and the western part.

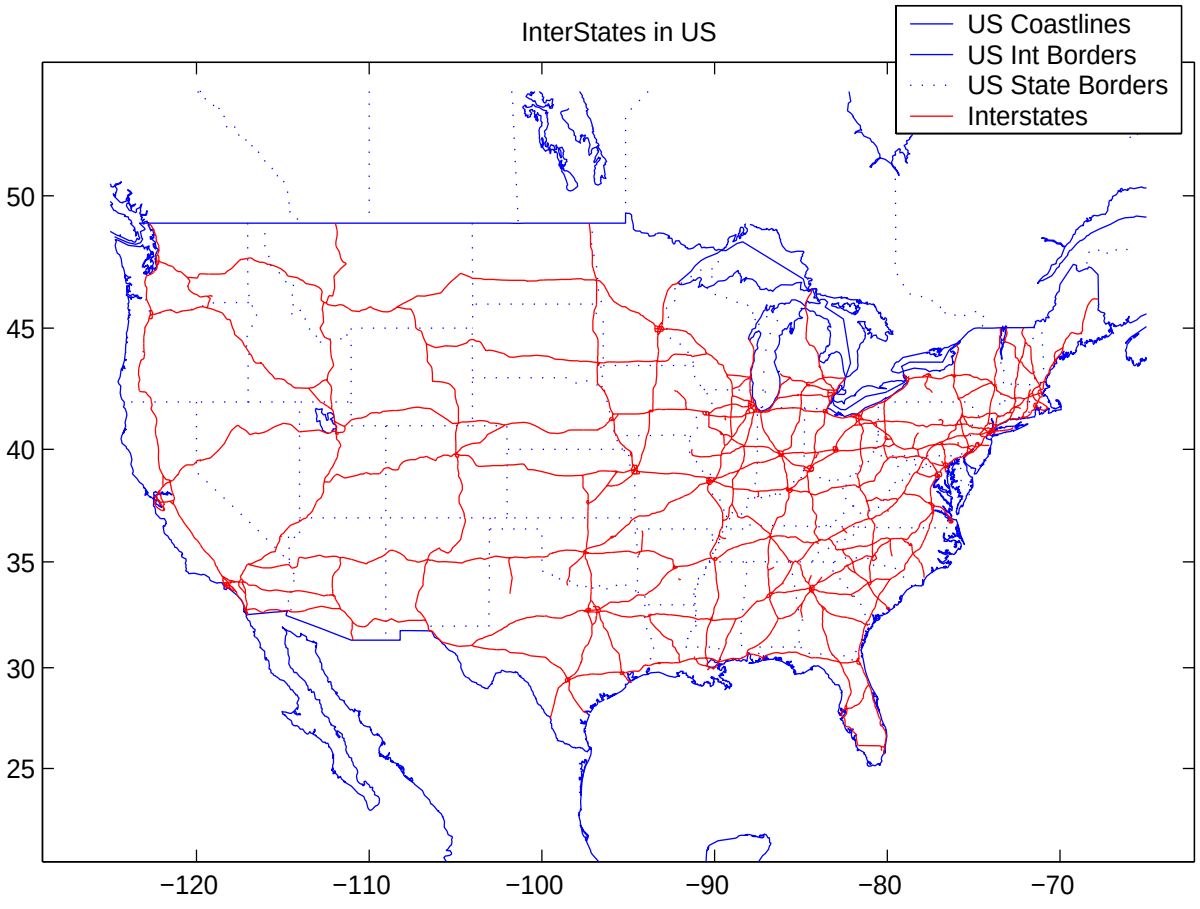


Fig 2.1 Interstate highways in U.S.

Therefore, some other roads need to be included to make it a complete uniform network, which can then be used as the underlying road network for our analysis. So, it is decided to include U.S. highways because these are the second fastest moving roads in the country after interstate highways and since most of the transportation will be done by trucks that are allowed on almost all of the U.S. highways. The following Fig 2.2 shows the entire U.S. highway network.

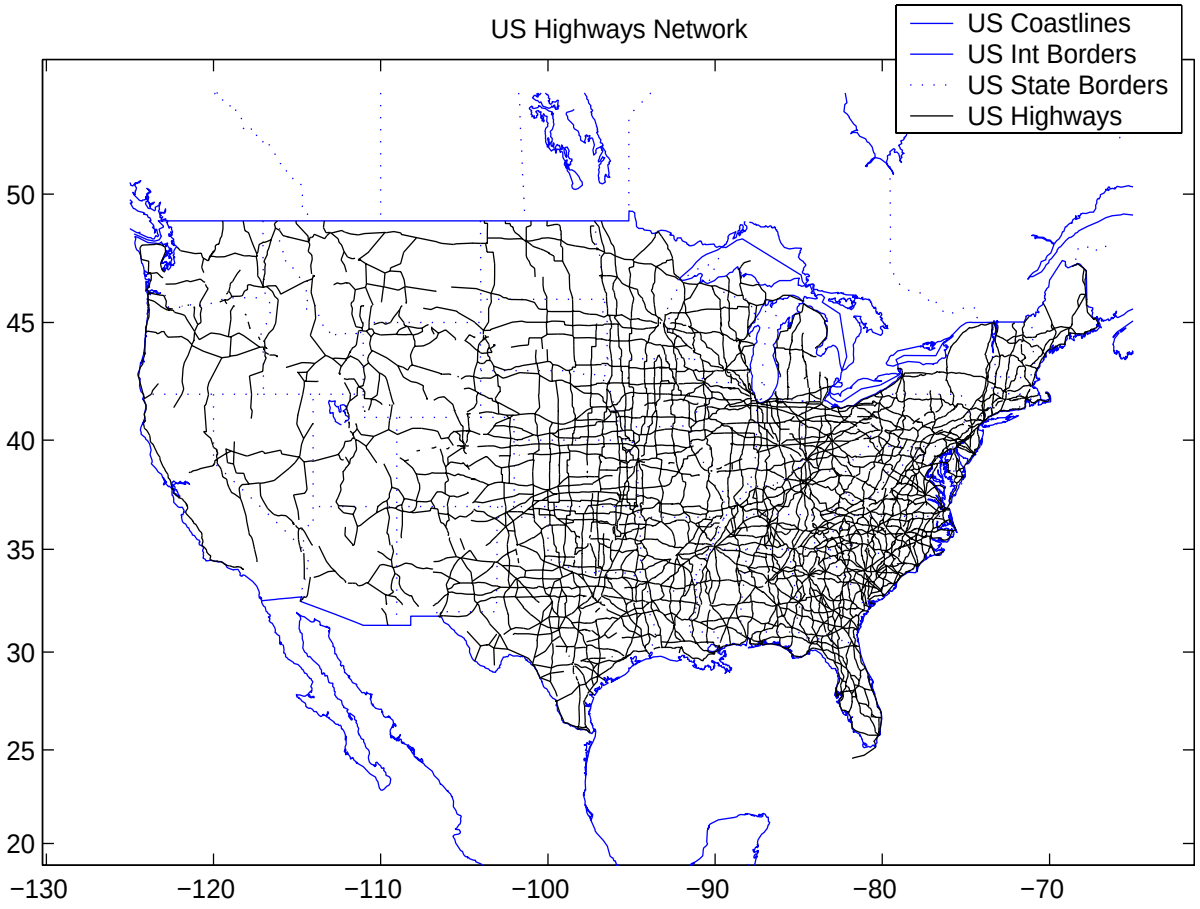


Fig 2.2 U.S. highways

Therefore, it is clearly seen that the entire U.S. highway network is very dense and it is not desirable to use the entire U.S. highways in the underlying road network. So, a method has to be developed to choose only part of the U.S. highways along with the entire Interstate

network to serve our purpose. Different heuristics are tried to get a good underlying road network that can be used to place DCs at the node intersections to make a complete public logistics network.

2.1 Heuristics to Design Underlying Road Network

A network or a graph is represented by *XY* and *IJD* matrix as follows:

[*XY*] = *longitude and latitude* coordinates of a node. ($n \times 2$ matrix with n as the number of nodes in the network.)

[*IJD*] = ($m \times 3$ matrix with m is the number of arcs in the network).

I → Index of first node (one end of an arc) in XY matrix,

J → Index of second node (other end of an arc) in XY matrix,

D→ Distance between nodes *I* and *J* (length of an arc) in miles.

The subroutine for developing underlying road network is as follows:

[*XY1*, *IJD1*] = *subgraph* (Only Interstates from whole U.S. Road Network)
[*XY2*, *IJD2*] = *subgraph* (Only Interstates and U.S. highways from entire U.S. Road Network)

"*subgraph*" is a function (developed by Dr. Kay [6]) that creates subgraph from a graph or extracts a smaller network from a big network.

[*XY11*, *IJD11*] = *thin* (*XY1*, *IJD1*); → *InterstateNet*
[*XY22*, *IJD22*] = *thin* (*XY2*, *IJD2*); → *IntstUSNet*

"*thin*" function thins the degree-two nodes from graph [6]. There will be a node only at the intersection of two roads or at the end of a road. So, a new network is obtained with a lesser number of nodes and arcs while keeping all of the necessary properties of the network.

Now there are two networks, one that has both interstates and U.S. highways (*IntstUSNet*) and the other that has only Interstates (*InterstateNet*).

2.1.1 Method I

The complete *InterstateNet* is needed to include all the interstate highways in the underlying road network and part of U.S. highways will be extracted from *IntstUSNet*. In method I, every node of *InterstateNet* is tested and some nodes are chosen around it to get a part of U.S. highways.

These nodes are chosen in the following way:

First, we take a rectangle with ± 8^o latitude and longitude (~550 miles) around this node and choose all the nodes in the *InterstateNet* that lies in this area. Then count the number of nodes in this area. If this number is more than a specified number called *NoNodes* = 8 nodes, then the area around this node is reduced to quarter of the previous area with ± 8/2 = 4^o latitude and longitude region around this node, and then keep reducing the area like this until the chosen number of nodes is less than *NoNodes*. This is done because more number of interstate nodes unnecessarily makes the algorithm slow and does not help in determining the part of U.S. highways to add to the network.

Fig 2.3 shows a part of *InterstateNet* with thinned road network in dark lines. There is a node at each intersection of this thinned network. Smaller dashed box shows the chosen area around the *Cnode* (test node) as a result of the larger box having more than *NoNodes* = 8 nodes.

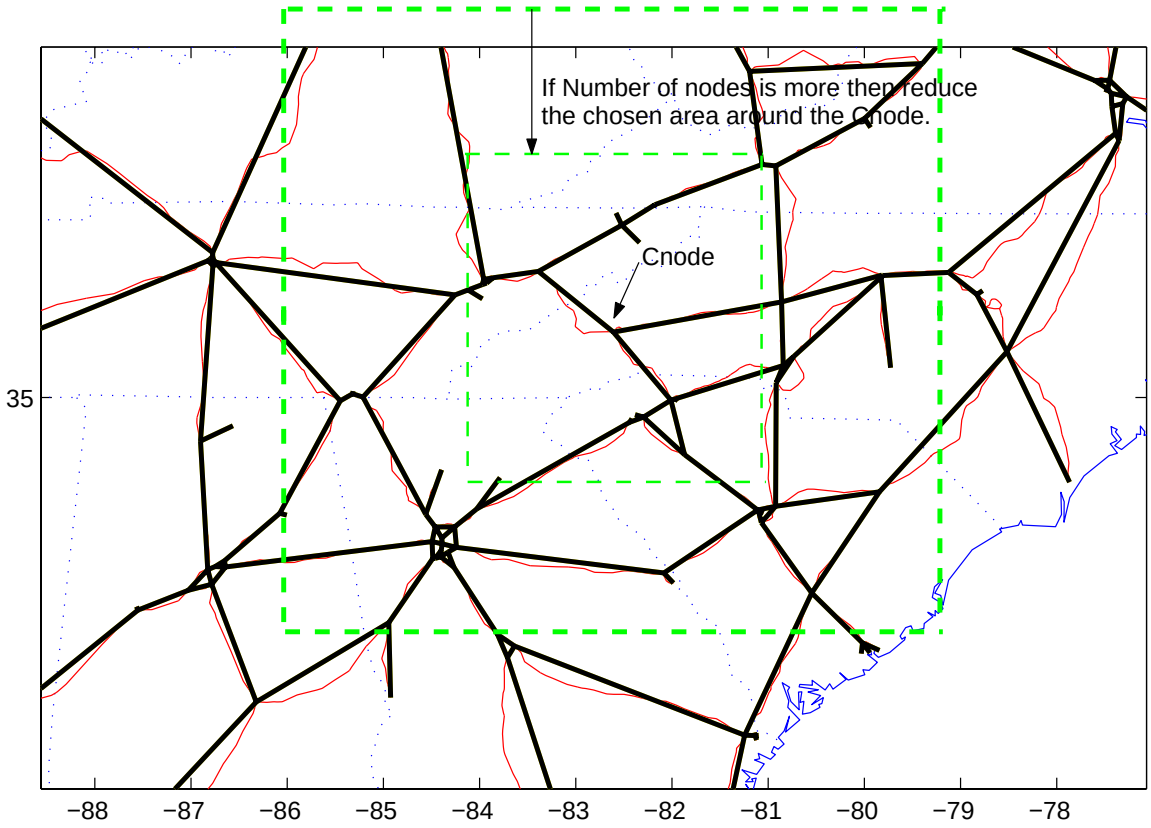


Fig 2.3 Part of Interstate Network with *Cnode* at the center and the area chosen around it to add U.S. highways to this network.

Now the distance between the *Cnode* and the chosen nodes around it called c_1, c_2, c_3,…,c_n is calculated.

Let D_1 be the shortest distance between the *Cnode* and c_1 in *InterstateNet* and let D_2 be the shortest distance between the *Cnode* and c_1 in IntstUSNet that consists of both interstates and U.S. highways. If D_2 is less than $x\%$ of D_1 then all the arcs composing shortest path between *Cnode* and c_1 in the IntstUSNet are included in the underlying road network. Dijkstra's algorithm is used to find out the shortest distance and path between two nodes in the network [6]. Different percentages of x were tried in this case for getting a good part of U.S. highways but the attempt to capture a large number of arcs was unsuccessful as a result method II was tried. It is discussed in the following section.

2.1.2 Method II

Method II is same a Method I except that the value of the differences between D_1 and D_2 was used as a parameter to decide the parts of the U.S. highways to be included in the final network instead of the percentage reduction. To develop the underlying road network different values of the difference in miles threshold were used.

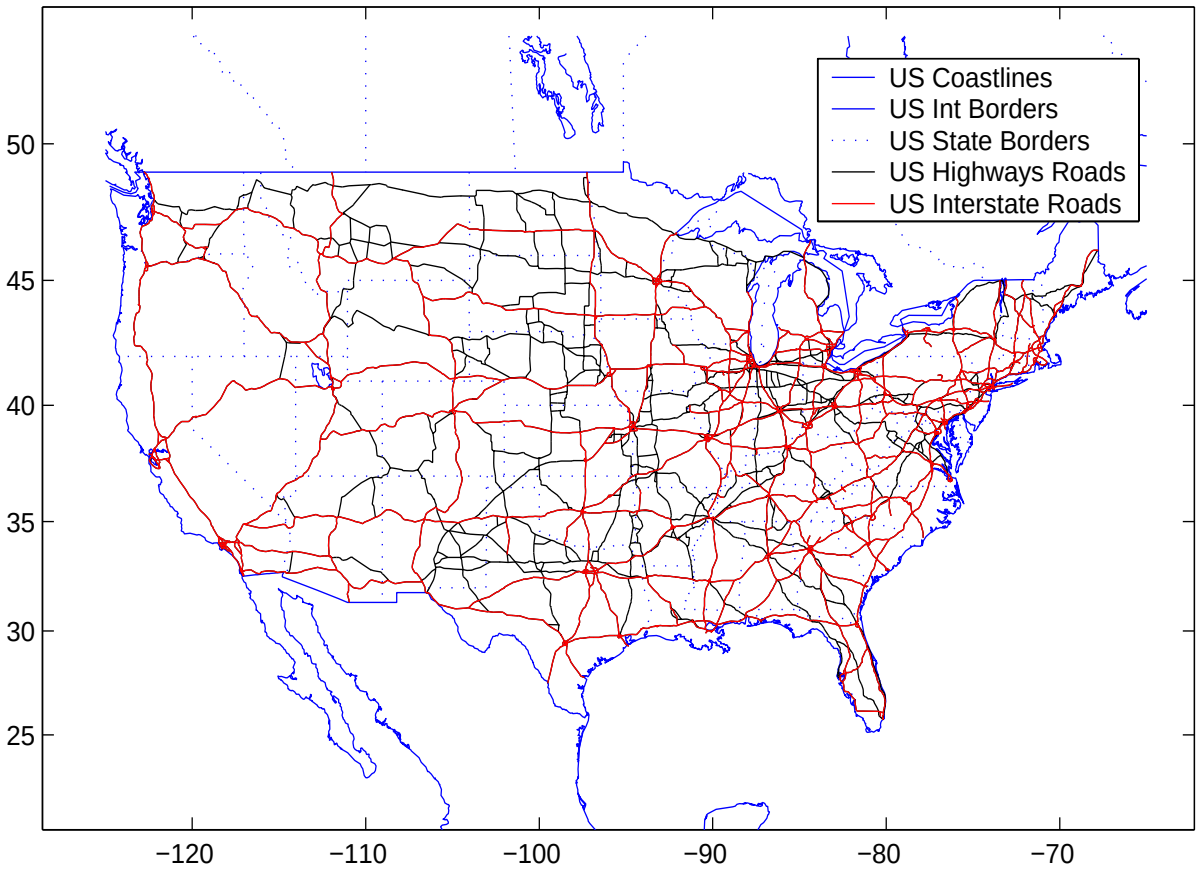


Fig 2.4 Interstates and U.S. highways Road Network obtained by method II

2.2 Addition to Above Heuristic

The underlying road network was obtained as shown in Fig 2.4 but in western part of the country there were not enough U.S. highways in the road network and there were too many in the portions of the eastern part of the country making it more dense which is not good for the analysis. All the U.S. highways in 3 states of the country viz. Nevada, California and Oregon are added to the network. Add them in the *InterstateNet* and *IntstUSNet* in the process of selecting U.S. highways by the above heuristics. The road network for U.S., shown in Fig 2.5, is more evenly distributed.

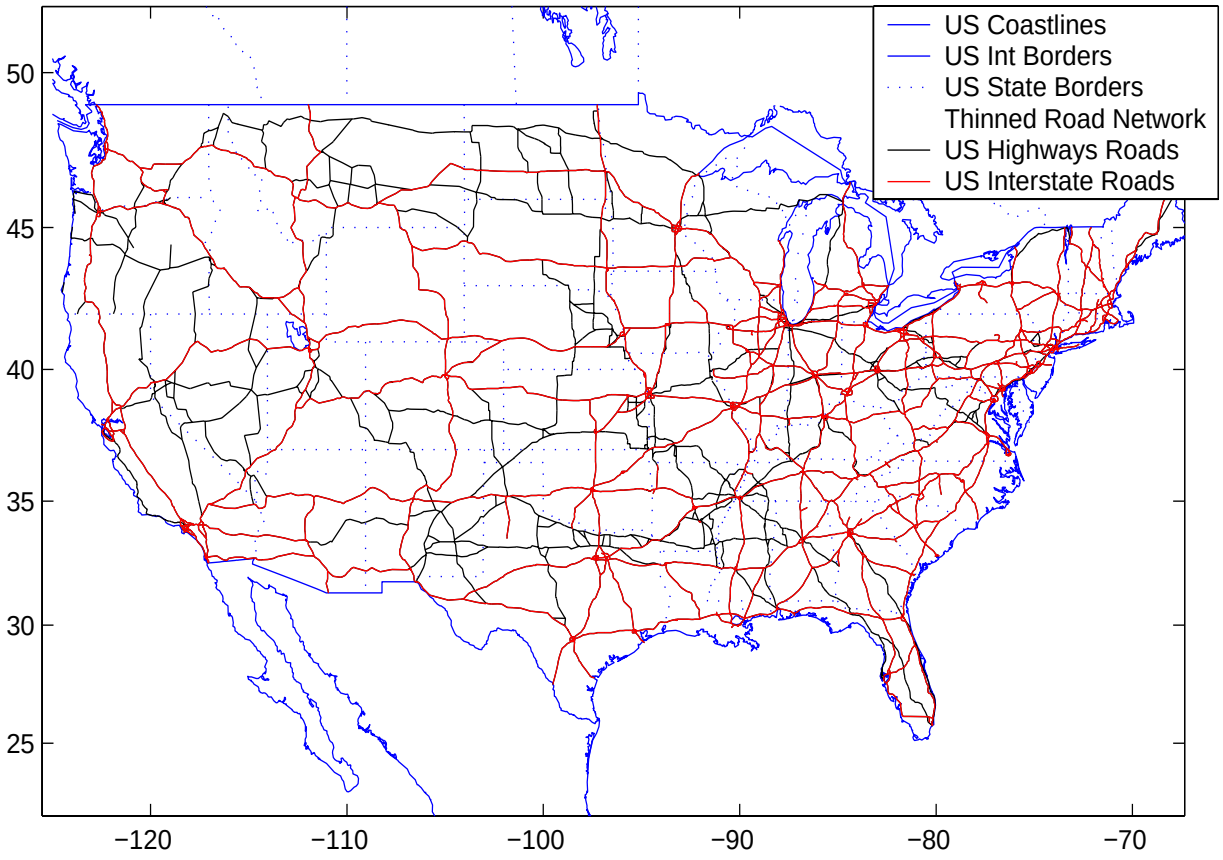


Fig 2.5 Underlying Road Network when parameter value of (D_2 - D_1) is 140 mi

2.3 Removing Unconnected Nodes

It is found out later in the analysis that some of the arcs in the above road network are not connected to other parts of the network. This number is not too large, but it causes problems and generates error in the heuristic. So the function named *RemoveUnconnNod*es [Appendix 1A] is developed which takes final network *IJD* and *XY* as an input and removes all the unconnected nodes and returns a network with all connected nodes.

This function first finds out the shortest distance between all the nodes in the network using Dijkstra's algorithm and then finds out the node sets which are not connected or where the distance is infinity. Eventually it finds the nodes that are not connected to the major part of the network. Remaining nodes are subgraphed to obtain a complete connected network.

2.4 Method III

Method III is an entirely different approach as compared to the above heuristics with the advantages of requiring less parameters. In this method, all cities with more than certain population are made targets. All of these selected cities are the sources used to generate demand for the road network. Delaunay triangulation [6] of these cities is done. Delaunay triangulation gives a set of triangles such that no data points are contained in any triangle's circumcircle. The domain is partitioned into small pieces (triangles). Edges of the triangle determine the cities to be joined directly by the roads.

If there is an edge between two cities (nodes) then shortest distance route is found between these two cities using Dijkstra's algorithm. The idea behind this approach is to join all the cities with population greater than certain population limit directly by shortest distance routes. This is done because in the public logistics network; generation of demand is directly proportional to population in an area. Highly dense areas like the cities with maximum population should be directly connected in the network and there should be a node or a

possible site of DCs in these areas. Fig 2.6 shows the road network obtained by this approach by joining cities of population greater then 20,000.

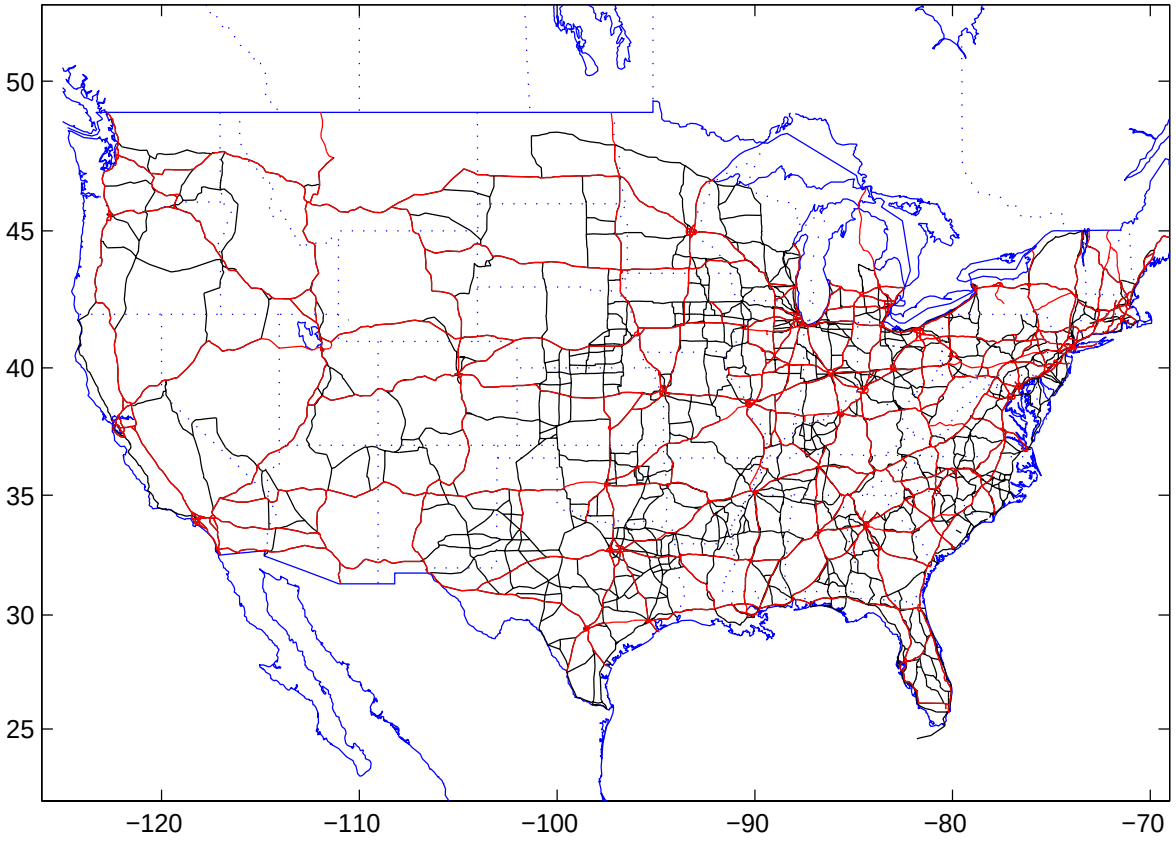


Fig 2.6 Underlying road network with joining cities of population 20,000 or more

It can be seen clearly from the above figure that some of the places are dense in the network. To reduce the number of roads in this network, every thinned node in this network is tested to be included in the final network. Shortest distance paths between all the nodes are found. Then using the proximity factor, which will be described in chapter 4, the actual demand was calculated for each node and then some of the nodes were discarded depending on threshold of volume of packages at each node. The network obtained by using this heuristic of joining cities of population greater then 20,000 and then removing arcs below a certain weight threshold results in a lot of discontinuities, 1-degree arcs, and small triangles. The small

triangles could be removed by third-degree thinning. In third-degree thinning, smaller triangles with DCs at each vertex are collapsed into a single DC placed at one of the vertices.

```
1. [XY, IJD] = subgraph (Only Interstates and U.S. Highways from entire U.S. Road Network);
2. [tIJD,idxIJD] = thin(IJD); % Thin degree 2 nodes.
3. [stXY,stIJD] = subgraph(XY,[],tIJD);  % Subgraph the thinned network.
4. [stXY,stIJD] = RemoveUnconnNodes (stXY,stIJD); % Remove the nodes which are not connected to
the network.
5. cityXY = uscity10k ( Choose cities with population greater then 20k).
6. idx=argmin(dists(stXY,cityXY));    % Indexes of nodes to which uszip5 is assigned.
7. NcityXY=stXY(idx); % closest nodes to the cities.
8. T = delaunay(NcityXY); %Delaunay triangulation of selected nodes.
9. IJ = tri2list(T); % Convert triangle indices to arc list representation.
10. for i=1: size(IJ,1)
11.       [d2,p2]=dijk(A,citys(IJ(I,1)),citys(-IJ(I,2))); % Find the shortest path between nodes.
12.       padd=[padd p2];  % Collect all nodes used in shortest path.
13. end for
14. [Xyfinal,IJDfinal] = subgraph(selected nodes 'padd' from XY,IJD network);
15. [IJC11,IJC12,IJC22] = addconnector(cityXY,Xyfinal,IJDfinal); % Connects all the cities to the
                                                                  nearest nodes.
16. makemap(Xyfinal);
17. pplot(IJDfinal,Xyfinal,'k-','Tag','Thinned Road Network')
18. pplot(cityXY,'g.')
```

Fig 2.7 MATLAB Pseudo-code for developing underlying road network

- RemoveUnconnNodes: Removes the nodes that are not connected to the whole network [Appendix 1A].

3. Developing a PLN and Estimating Its Average Travel Time.

The previous chapter described the development of the underlying road network. Now in this chapter, a public logistics network (PLN) will be developed using this underlying road network. A parameter called the average travel time will then be estimated and will provide a basis for comparing different PLNs and choosing the best. The procedure used for estimating the average travel time in PLN is based on Kay and Parlikad [5].

3.1 Developing a PLN

Developing a PLN consists of several steps including choosing the DCs, connecting new arcs in the network, removing some arcs from the network and then determining the population assigned to each DC based on the average weighted distance of the five digit us zip locations closest to that DC.

3.1.1 DCs

Every intersection point in the underlying road network can act as a distribution center (DC) location. Therefore, the first query is to determine which intersection point of roads or nodes in the network should be chosen as DCs and how many such DCs should be chosen. The number of DCs is chosen by using a parameter called *p*, which is the probability of choosing any node depending on its weight (or the population assigned to it).

The nodes at which DCs will be located are selected as follows:

Let n be the number of nodes in the network.

1. Generate n random numbers between 0 and 1. Call them $r_1, r_2, \ldots, r_n$.
2. Divide the weight of each node with the average of the weights of all the nodes and call these numbers $a_1, a_2, \ldots, a_n$.
3. Choose node i if $r_i < pa_i$, $i = 1 \ldots n$.

These nodes will be the selected nodes that will act as DCs in the network. The greater the population around a node, the greater is its chance of getting selected. Therefore, the chance of selecting a DC is directly proportional to the population around it [2, *wtbinselect.m*].

3.1.2 Public Logistics Network generation

Once the DCs to be connected are found, generation of a public logistics network involves following four steps:

1. The pairs of specified points that are considered to be adjacent are identified by Delaunay triangulation of the points. Each edge in this triangulation defines an adjacency. Figure below shows the Delaunay triangulation of the cities with population greater than 100,000.

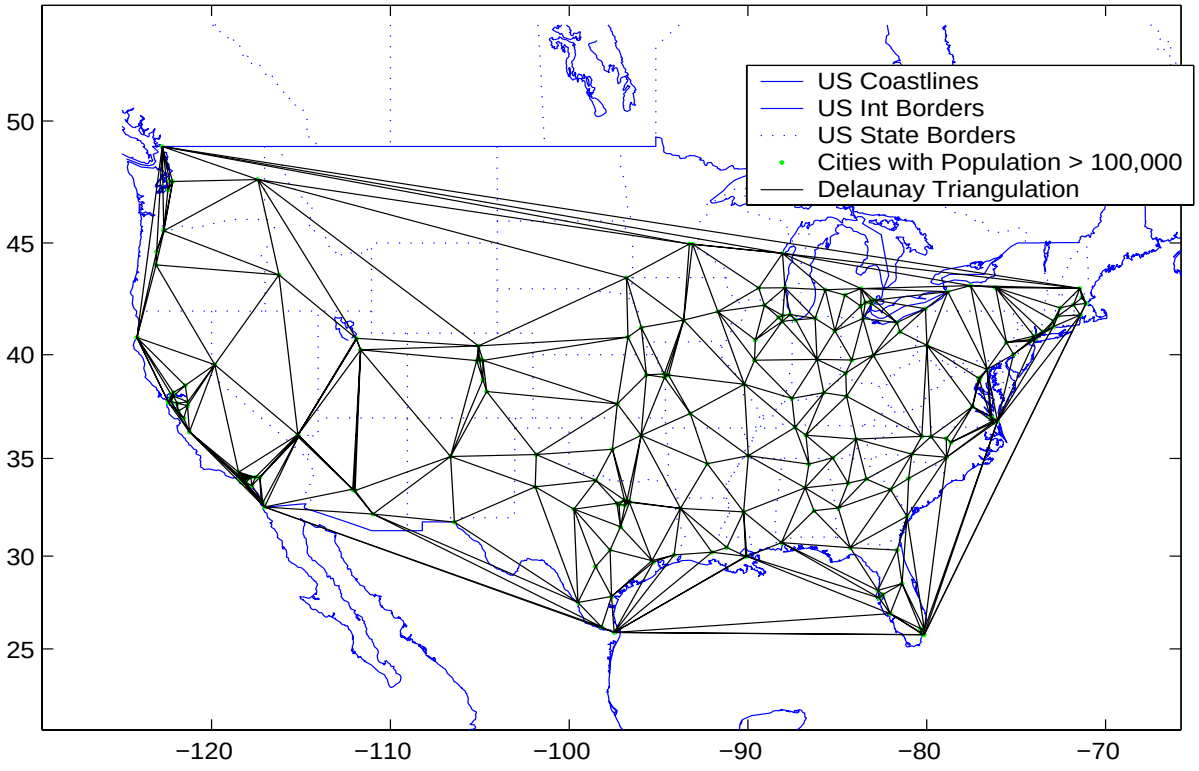


Fig 3.1 Delaunay triangulation of U.S. Cities with population 100K or more

Delaunay triangulation gives a set of triangles such that no data points are contained in any triangle's circumcircle. The domain is partitioned into small pieces (triangles). Delaunay Triangulation maximizes the minimum angle of all the angles in the triangulation. It avoids triangles with small angles.

2. The optimal (minimum-distance) path through the region between each pair of adjacent nodes is calculated. Dijkstra's shortest-path algorithm is applied to all the pre-specified adjacent node (DCs) pairs to determine the shortest distance paths.

3. The selection of which of these arcs to use in the final network is made, resulting in a road network that connects all the pre-specified points. For selecting these arcs, the following method is adopted:

Every triangle is tested for removal of arcs. If the largest side (arc of the network) of the triangle is less than some threshold of the sum of the other two sides then this arc is deleted from the final network. Fig 3.2 describes the above approach in detail.

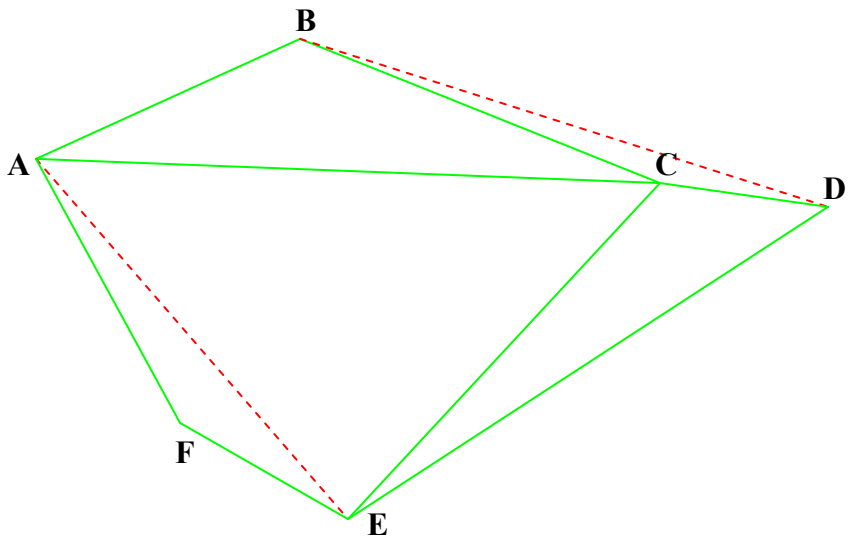


Fig 3.2 Removal of Arcs from the network—dashed lines show the removed arcs from the network.

In Δ *BCD*, the largest side of the triangle, i.e., *BD* is more than some threshold value (say 90% as discussed in next chapter in detail) of the sum of the length of other two sides of this triangle then the arc *BD* is removed from the network. So now if something has to go from *B* to *D*, it will go from *B* to *C* first and then from *C* to *D*.

In Δ *ABC*, the largest side is less than the threshold criterion so no arc is removed from this triangle and all the arcs in this triangle will remain in the final network.

The idea behind removing this arc is to reduce the number of direct arcs in the network and to reduce the average waiting time of a package. For example, If a package has to go from *B* to *D*, it will go from *B* to *C* first and then from *C* to *D*. Because of this, if there was only one truck running from *B* to *C* and *C* to *D*, then there is a possibility that two trucks need to run

now from *B* to *C* and *C* to *D* because of the increase in demand of that route. It leads in the reduction of waiting time of the truck and hence improving the overall average travel time of the package.

4. Some more paths in the network are added in this stage. The adjacent triangles or the triangles sharing the same edge are chosen. Every such pair of adjacent triangles is tested for addition of arcs in the network.

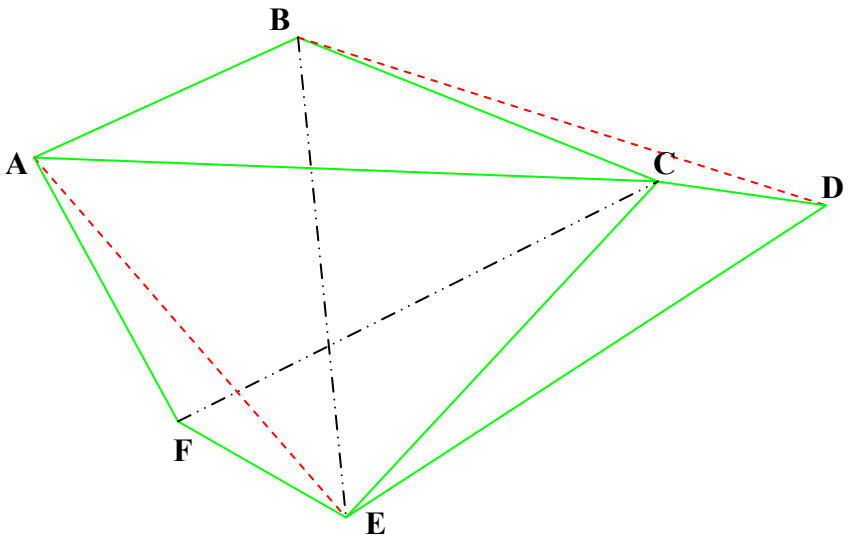


Fig 3.3 Addition of arcs in the network—black double dotted dashed lines show the new additions to the network.

For example, Δ *ABC* and Δ *ACE* are neighbor triangles sharing the same edge *AC*. The shortest distance between nodes *B* and *E* in the initial network and the shortest distance between the same nodes in the reduced network obtained after the Delaunay triangulation of nodes is calculated. If the direct distance between the opposite nodes of the adjacent triangles in initial complete network is less than some threshold (discussed in next chapter in detail) of their shortest distance in the new reduced network, then this arc is added to the new network which is shown by double dotted dashed line joining *B* and *E* in Fig 3.3.

Now this final arc network will be evaluated for finding the average travel time of the packages in the network. Final network is obtained as shown in Fig 3.4 and Fig 3.5. This procedure is implemented in *AverageTravelTime.m* in the Appendix 1A.

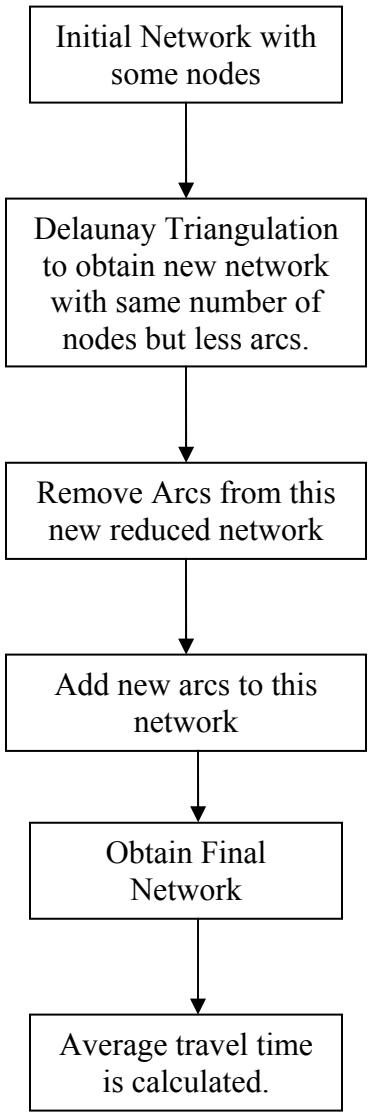


Fig 3.4 Flowchart shows steps involved in getting a public logistics network that will be evaluated for average travel time.

```
1.  Load XY, IJD (Underlying Road Network); % Created in last chapter
2.  WT      = InitWt(XY); % Calculates population assigned to each node.
3.  d0      = dijkstra(IJD); % Shortest distance between each pair of nodes in network.
4.  p       = 0.01;
5.  nXY     = XY(find(rand(1,length(WT)) < p * WT/mean(WT)),:); % Randomly select DCs.
6.  T       = delaunay(nXY(:,1),nXY(:,2)); %Delaunay triangulation of selected nodes.
7.  IJ      = tri2list(T);
8.  xIJD    = Connector(T,b,d0,threshA,c);
                                        % Calls the connector function which returns the
                                        new arcs that need to be added to this network.
9.  rIJD    = RemoveArcs(T, b, threshR, c);   % Returns the IJD matrix of the arcs to be
                                                removed from the new network.
10. DC = PopAndPos(nXY);    % Return a structure with XY coordinates, population and avg.
                                weighted distance.
11. IJD     = IJD + xIJD - rIJD ;
```

Fig 3.5 MATLAB Pseudo-code for developing PLN

- *InitWt*: Finds the U.S. population distribution for each node in the network [Appendix 1A].

- *Connector*: Finds the new arcs to be added to a reduced network obtained by Delaunay triangulation of the original network [Appendix 1A].

- *RemoveArcs*: Find the arcs to be removed from a new network obtained by the Delaunay triangulation of the selected nodes from the original network [Appendix 1A].

- *PopAndPos*: Returns a structure with Position (longitude and latitude coordinates) of the nodes, U.S. population distribution for each node and the average weighted distance of a node to the 5-digit zip code locations closest to this node [Appendix 1A].

3.2 Estimating Average Travel Time

In this section average travel time is estimated using the total demand in the network as estimated by Kay and Parlikad [5] taking the proximity factor into consideration.

3.2.1 Transport Demand

The average daily demand of 10.434 million packages handled by UPS throughout the United States [8] is used as the basis of determining a representative range of likely package demands for the region.

UPS demand is used because the type of packages handled by UPS (e.g., less than 150 lbs.) is similar to the type of packages envisioned to be handled by the proposed public logistics network. The U.S. Census Bureau publishes a Commodity Flow Survey [7] every five years that tracks all parcel, U.S. Postal Service, and courier shipments. This data is not used because it only includes the tons shipped, not the number of packages shipped [8].

3.2.2 Proximity Factor

Transport demand to and from each pair of DCs is estimated by using the population percentages of each DC together with a proximity factor that controls the degree to which a

DC is more likely to transport packages to nearby DCs as opposed to DCs located further away.

Let ω_i be DC_i's percentage of the total population. Without a proximity factor adjustment, the transport demand between DC_i and DC_j is $\omega_{ij}^{\ 0} = \omega_i \omega_j$ and $\omega_{ii}^{\ 0}$ is the demand within the region covered by DC_i. Given m DCs, $DC_{[1]}$, $DC_{[2]}$, …, $DC_{[m]}$, ordered in terms of their increasing great circle distance from DC_i, a proximity factor of pf is used in a normalized geometric distribution [5] as follows:

$$\omega'_{i[j]} = \omega^{0}{}_{i[j]} \cdot \frac{\frac{pf}{m}\left(1-\frac{pf}{m}\right)^{(j-1)}}{\frac{1}{m}\cdot\sum_{k=1}^{m}\frac{pf}{m}\left(1-\frac{pf}{m}\right)^{(k-1)}} = \omega^{0}{}_{i[j]} \cdot \frac{\left(1-\frac{pf}{m}\right)^{(j-1)}}{\frac{1}{m}\cdot\sum_{k=1}^{m}\left(1-\frac{pf}{m}\right)^{(k-1)}}$$

$$\omega_{i[j]} = \frac{\omega'_{i[j]}}{\sum_{k=1}^{m}\sum_{l=1}^{m}\omega'_{kl}}$$

Both $\sum_{i=1}^{m}\sum_{j=1}^{m}\omega^{0}{}_{i,j} = 1$, $\sum_{i=1}^{m}\sum_{j=1}^{m}\omega_{i,j} = 1$.

The proximity factor orders DCs in terms of their distances. The proximity factor for this model is estimated using the real data of actual commodities shipped from state to state and will be described in detail in next chapter where a nonlinear regression is used to estimate the proximity factor.

3.2.3 Transport Time

The total time taken to transport a package from DC_i to DC_j, t_{ij}, is modeled as the sum of its local travel time to DC_i and from DC_j, its travel time on each truck between DCs, its loading and unloading time at each DC visited, and the time spend waiting at each DC for an available truck:

$$t_{ij} = \text{Local Travel Time} + \text{Travel Time} + \text{L/U Time} + \text{Wait-for-Truck-Time}$$

The route selected for transport between DCs is the one that minimizes the sum of the travel and L/U times; wait-for-truck time is not considered.

Truck waiting time is estimated by first summing, for each link in the network, the total DC-to-DC demand that is transported over the link and then diving this by the average truck load to get the average (fractional) number of truck trips needed for the link. Then, assuming that both the trucks and packages arrive at random (i.e., a Poisson process, or exponential inter-arrival times), the average time between trips is used as the expected waiting time for any package traversing the link (e.g., 3 trips per day would imply an 8 hour waiting time) [5].

The average truck load is estimated multiplying the maximum truck capacity by an average load factor (0.80). This corresponds to the assumption that the average truck is 80% full when it traverses a link, which implies that most packages have to wait for a truck. This wait is assumed to be significantly more than any of the other possible delays.

The average transport time of PLN network can be determined for a variety of different parameter values. The parameter values are summarized in the following Table 3.1.

Table 3.1 Parameters value used in estimating average travel time.

Parameter Name	Variable Name in Pseudo-Code	Values
UPS daily demand	Demand	10.434 million packages
Average load factor of truck	Lfactor	0.8
Maximum load capacity of Truck	Tload	120
Proximity Factor	w	6.554
Percent of UPS demand	Dfactor	100%
Loading/Unloading Time	LUtime	5,10 min
Local Truck speed within the DC area	TspeedLocal	45 mi/hr

The effects of these parameters will be studied in detail in the next chapter.

The average transport time is determined as follows:

$$Average\,Transport\,Time = \sum_{i=1}^{m}\sum_{j=1}^{m} \omega_{i,j}\, t_{i,j}$$

ω_{ij} = proximity factor between cities *i* and *j* and

t_{ij} = shortest route time between cities *i* and *j*.

```
1.  Load DC, IJD, XY  % PLN from section 3.1
2.  n = size (DC);   % No. of nodes in the network.
3.  distance      = dists(DC.XY);   % Distance between all the nodes.
4.  w1            = DC.Pop/sum(DC.Pop);  % Pct. of total population.
5.  w             = proxfac(distance,w1);  % Using the proxfac function to find the order based
    proximity factor.
6.  IJD(:,3)      = [IJD(:,[3])+2*LUtime ]; % Adding Loading/Unloading time to the arcs.
7.  A             = list2adj(IJD);
8.  [d0,p0]       = dijk(A);    % Shortest route distance and paths between the nodes
9.  Ttime         = d0 + (Avg. weighted distance between DC and all U.S.ZIP5 locations covered by
    it)*1.2/TspeedLocal;
10. wnew          = w*Demand*Dfactor/(Tload*Lfactor);
11. for i=1:n, j=1:n
12.       vol(each node)=vol(each node)+wnew(i,j); % vol is the total demand at each node.
13.  end for
14. wtime         = 24./vol; % Truck waiting time in a link(arc).
15. for i=1:n, j=1:n
16.            WFTT(i,j)= sum(diag(wtime([p{i,j}(1:end-1)],[p{i,j}(2:end)]))); % Wait for  Truck time
    between any 2 DC's.
17. end for
18. Totaltime     = WFTT + Ttime;   % Total time = Travel Time [Within DC's(only at start and end)
    and from one DC to another DC] + L/U time + Wait for Truck Time
19. avgTtime      = sum(sum(Totaltime.*w));    % Average Travel Time.
```

Fig 3. 6 MATLAB Pseudo-code for estimating average travel time

- *proxfac*: It calculates the order-based proximity factor using the formulas in section 3.2.2 [Appendix 1A].

4. Parameter Selection

Different parameters are used to design the underlying road network, to select the DCs in the network, to develop the public logistics network, and finally to calculate the average travel time of packages in the network. Since some of these parameters can be very sensitive in obtaining the road network, in generating public logistics network and might affect the final solution. They are studied in this chapter and their effects on the final solution are determined.

4.1 List of Parameters

Table 4.1 shows the list of parameters used to develop underlying road network (URN), public logistics network (PLN), their values, sensitivity (in determining the size of the network) and the number of nodes and arcs in the network generated using these parameters. Sensitivity analysis is done in section 4.1.1.

Table 4.1 Parameters used in developing URN, PLN

Parameter Name	Where used	Value	No. of Nodes	No. of Arcs	Sensitivity
Min. population size of the cities	URN (sec 2.5)	20,000	2956	4616	High
Difference of the shortest distance in InterstateNet and IntstUSNet (D_2 - D_1)	URN (method II)	100 mi	1805	2778	High
Max number of nodes to check around test node	URN (method I, II)	8	1805	2778	Low
Shortest distance between nodes to consider it for analysis	URN (method I, II)	50 mi	1805	2778	Low
Ratio of shortest distance in complete network and reduced network	PLN (Addition of arcs)	0.85	N/A	N/A	Low
Inverse of the ratio of the largest side in the triangle to sum of other two sides	PLN (Removal of Arcs)	1.1	N/A	N/A	Low

Table 4.2 shows the assumptions on the speed of various roads. Data of the road distances and their parameters are obtained from the Oak Ridge National Highway Network [2, *usrdlink.m*].

Table 4.2 Assumptions on the speed of roads:

Road Type	Speed (mi/hr)
Interstate & Rural	70
Interstate & ~Rural	60
Fully controlled & Rural & ~Interstate	55
Divided highway & Rural	50
4+ lanes & Rural	45
Rural	40
Fully controlled & ~Rural & ~Interstate	35
Divided highway & ~Rural	30
~Rural	25
All other roads	35

4.1.1 Effects of these parameters

In this section, the effects of the highly sensitive parameters that are used to develop the URN and the PLN are studied.

- Min. population size of the cities. Table 4.3 clearly shows that the size of the network will keep on reducing with increase in the parameter value of minimum population size of the cities.

Table 4.3 Number of nodes and arcs in the network

Min. population size of the city	No. of Nodes	No. of Arcs
10,000	4493	6912
20,000	3214	4882
50,000	1734	2571
100,000	938	1380
200,000	486	711
500,000	198	288

- Difference of the shortest distance, (D_2 - D_1), in *InterstateNet* (Network obtained by thinning of Only Interstates from whole U.S. Road Network) and *IntstUSNet* (Network obtained by thinning Interstates and U.S. highways from whole U.S. Road Network). Table 4.4 and Fig 4.1 show the size of the network obtained by using the different parameter values of the differences between D_1 and D_2.

Table 4.4 Number of nodes and arcs in network with different values of parameter (D_2 - D_1)

Distance (D2 - D1)	No. of Nodes	No. of Arcs
80	2093	3257
90	1985	3076
100	1805	2778
110	1708	2620
120	1653	2530
130	1624	2479
140	1553	2357
150	1539	2332
160	1461	2208
170	1380	2077
180	1319	1978
190	1296	1940
200	1288	1928

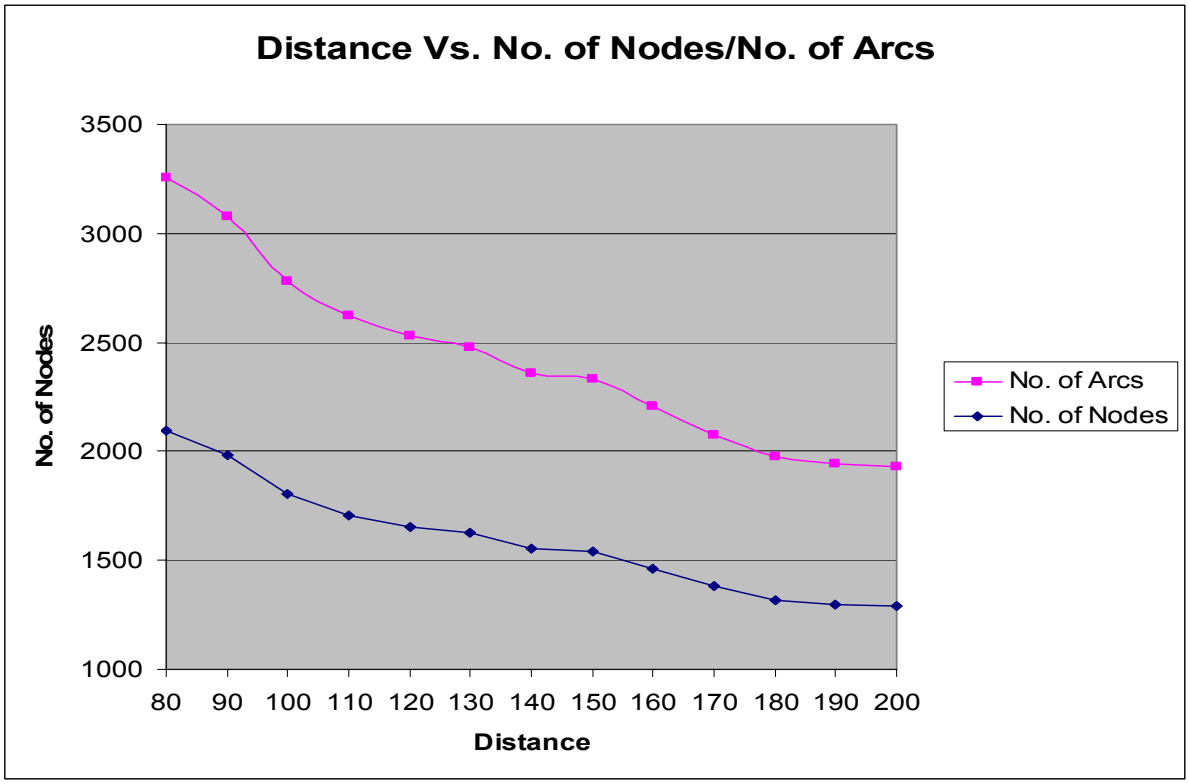


Fig 4.1 Distance Parameter (D_2 - D_1) vs. No. of nodes/arcs in the network

4.2 Checking the quality of the network

Several parameters are used to check the quality of the network. Some of these are as follows:

- The distance of each node in the network to its closest node and the distances of each DC to its closest DC in the network.

For the network obtained by method III by joining cities of population 20,000 or more as shown in the Fig 2.6, following figure shows the sorted distance of nodes to its closest node.

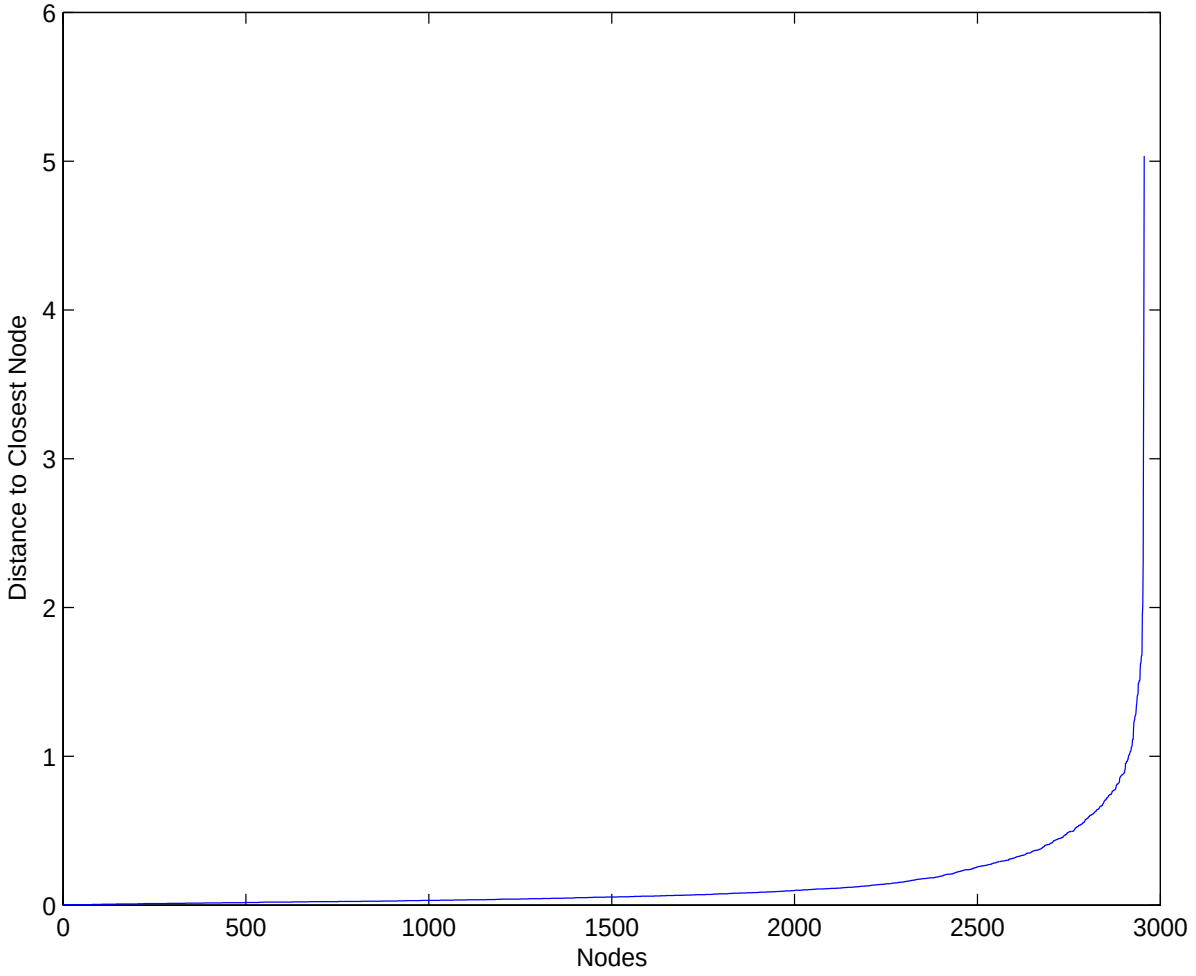


Fig 4.2 Sorted distances of the nodes to its closest node for the network in Fig 2.6

It can be clearly seen from the Fig 4.2 that the maximum travel between the nodes to its closest node is 5hr, which is a comparatively small number and a very good indicator of a network being evenly distributed in the entire area. Also, most of the nodes, i.e., 97%, require a travel of 1hr or less to its closest node.

Similarly for DCs, as shown in Fig 4.3, the maximum travel required is 6hr which is also a relatively small number and most of the nodes, i.e., 90%, require a travel of 2hr or less to its closest node.

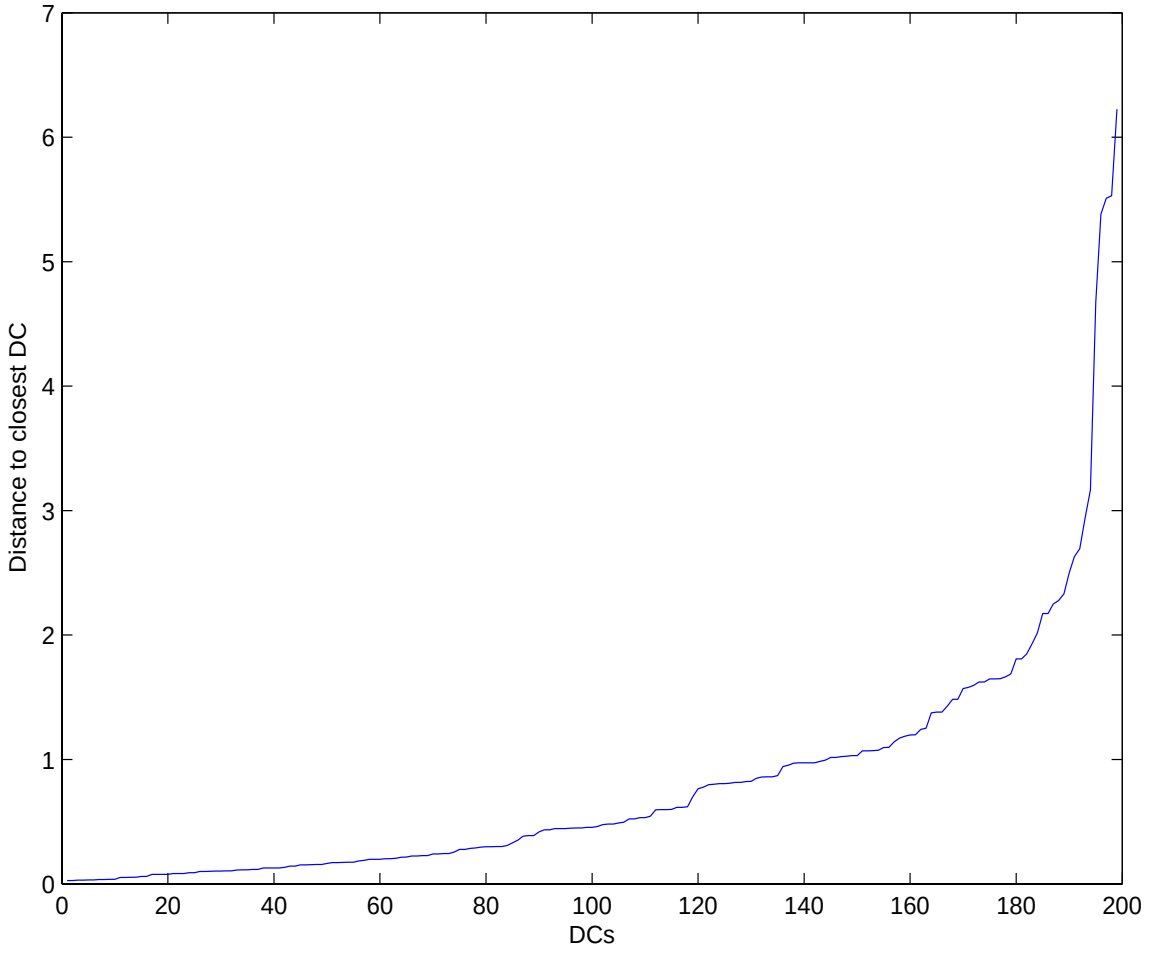


Fig 4.3 Sorted distance of 200 randomly chosen DCs to their closest DC in the network of Fig 2.6

- The average weighted distance of a DC to the 5 digit zip code locations closest to this DC.

For example, for the network in Fig 2.6, DCs are placed randomly on 200 nodes in the network and then the average weighted distance is calculated for each DC using the *PopAndPos* function [Appendix 1A].

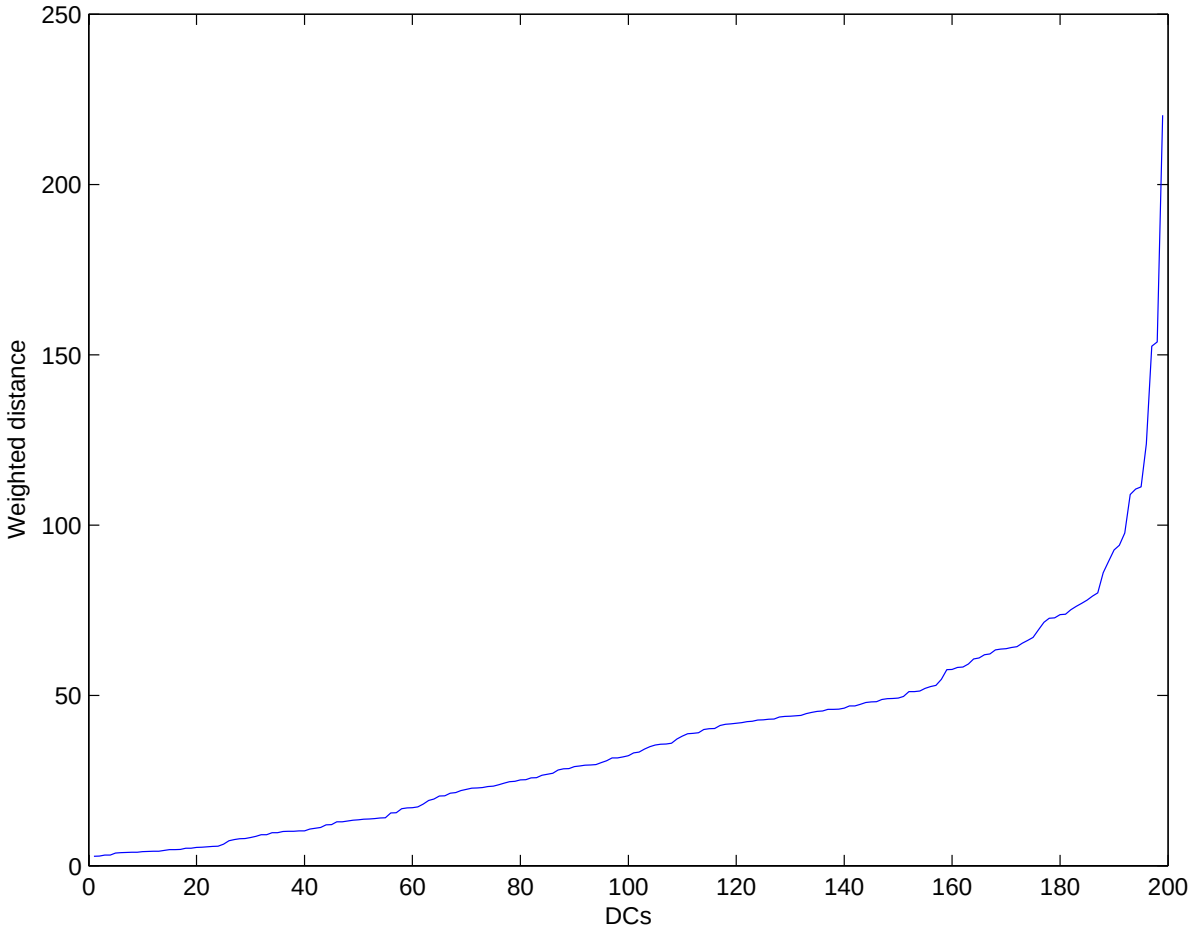


Fig 4.4 Weighted distances of 200 randomly distributed DCs in network of Fig 2.6

Fig 4.4 shows that the maximum value of average weighted distance is 220 miles in this network and the average is 37.5 miles, which indicates the quality of the network, and the uniform spread of the DCs in the entire area.

4.3 Estimating the proximity factor

The data of the number of packages shipped from one state to another published in U.S. Census Bureau, Transportation-Commodity Flow Survey 1997 [7] is used to get an approximate value of proximity factor.

Commodity flow data is used to calculate the value of proximity factor. Nonlinear least-squares data fitting by the Gauss-Newton method is used to fit this data [2, *nlinfit.m* function]. Population centroids for all the states are calculated. The XY coordinate/population centroid of a state is a weighted mean of coordinates of the cities within the state, the weights being determined by the population of each city. Then the direct distances were found between the centroids of the states by using the proximity factor calculation as described in section 3.2.2 and nonlinear least-squares data fitting. The proximity factor estimated is a good fit for the given data with 95% confidence interval of [6.4139, 6.6986]. Proximity factor is 6.55413213443378, which is used for all the calculations of average travel time in the network.

5. GA Experiments and Analysis

5.1 Genetic Algorithm

Genetic algorithms (GA) have been shown to be a promising approach for a wide range of search and optimization problems and particularly for the problems of allocation and relocation of DCs in the network as described in [4].

5.1.1 Model

There is n node undirected graph and DCs on $x \leq n$ of these nodes have to be placed. The problem is to minimize the average travel time of packages in the public logistics network generated by selecting x nodes from the graph.

5.1.2 Structure Representation and Initialization

Vector of n variables is used as the data structure, where n is the number of nodes in the network or the available sites for DC placement in the entire road network. A binary representation for the solution is used, where for each element of the vector

1 = DC is placed at the node.
0 = DC is not placed at the node.

x number of nodes are assigned with the value 1 in the initial solution which is decided as described in the section 3.1.1. This is based on the weight or the population assigned to each node in the network.

5.1.3 Evaluation and Fitness Scaling

The average travel/transport time as described in the section 3.2.3 is used as an evaluation parameter. Both the average travel time and the number of DCs will eventually be parameters in deciding the final solution of the problem.

5.1.4 Iteration and Population Size

Run the simulation for *n* number of iterations and then stop. Then select the best chromosome or the solution with the minimum average travel time as the best solution of DC locations for the PLN.

It was empirically found that a population size of 10 to 25 is a good number to run the GA simulation. This is because there is a lot of variability in the solutions of the problem and the number of DCs selected is only about 10% of the total available sites. There is enough variability in the population. In addition, with this population size the simulation runs very fast and more iteration can be done in short interval of time to get better results.

5.1.5 Reproduction

The "Roulette wheel" method is used to generate the offspring. Roulette wheel is the traditional selection function with the probability of surviving equal to the fitness of i_{th} individual divided by sum of the fitness of all individuals. These offspring construct a new population in which the population size keeps constant.

5.1.6 Crossover

Simple crossover is used. It takes two parents, P1 and P2, and performs simple single point crossover.

5.1.7 Mutation

Binary mutation is used which changes each of the bits of the parent based on the probability of mutation. Binary mutation probability of 0.2% is used.

5.1.8 Implementation

The *Genetic Algorithm Optimization Toolbox (GAOT)* [2] is used to run the simulation with some changes being made in the toolbox files to fit the model.

5.2 Improving the GA performance

After the initial few runs of the GA, it was observed that clusters of cities were forming in the solutions at certain places. The GA was finding it difficult to remove these clusters in a reasonable amount of time. To improve the performance of the GA and get a better solution, pre-processing of each solution is done. This pre-processing removes some DCs from the solution and ensures that the minimum distance between any two DCs is greater than a threshold of 25 miles. This value was chosen based on an ad hoc estimate of what the minimum separation between DCs should be.

Distance between all the DCs is calculated. If the distance between any two DCs is less than the threshold value, then one of them is dropped from the solution. The DC to drop depends on their weights, which is the population assigned to that DC calculated using the *PopAndPos.m* function [Appendix 1A]. The lower weight DC is dropped, repeatedly

removing DCs until the distances between DCs are all greater than 25 miles. This procedure is implemented in *PrepInitSol.m* in the Appendix 1A.

5.3 Results

Using the heuristics suggested in section 2.2.2, method II, the GA was run with initial population sizes of 10, 15, 20 and 25 chromosomes. The simulation was started with selecting a lesser number of nodes to serve as DCs in the network; the objective is to minimize the number of DCs along with getting the minimum average travel time in the network.

Table 5.1 shows the results of the GA simulation for a network generated by method II using parameter value of $(D_2 - D_1) = 100$ mi.

Table 5.1 Results of GA run using Method II.

GA run	Population Size (Max. No. of Iterations)	Best Solution (min Avg. Travel time)	No. of DCs
1	25 (40)	10.72 hrs	134
2	20 (50)	10.80 hrs	155
3	15 (67)	10.56 hrs	169
4	10 (100)	10. 51 hrs	184

Now, GA is run for network generated by method III, with different population sizes and different loading and unloading times of 5 and 10 minutes.

Table 5.2 shows the GA results when maximum load capacity of truck is 120 packages and loading/unloading time of 10 min.

Table 5.2 Results of GA run with L/U time of 10 min.

GA run	Population Size (Max. No. of Iterations)	Best Solution (min Avg. Travel time)	No. of DCs
1	25 (40)	10.50 hrs	173
2	20 (50)	10.67 hrs	207
3	15 (67)	10.43 hrs	198
4	10 (100)	10.32 hrs	183

Table 5.1 and 5.2 shows that the best solutions generated by method III are better then the solutions generated by method II. So, method III is explored in detail using different values of loading and unloading time.

Table 5.3 shows the GA results when maximum load capacity of truck is 120 packages and loading/unloading time of 5 min.

Table 5.3 Results of GA run with L/U time of 5 min

GA run	Population Size (No. of Iterations)	Best Solution (min Avg. Travel time)	No. of DCs
1	25 (40)	10.11 hrs	236
2	20 (50)	10.09 hrs	169
3	15 (67)	9.83 hrs	274
4	10 (100)	9.94 hrs	234

It was found from the initial few runs that the small population size of 10 was the best of the different population sizes chosen. Then the GA was run for one complete day and the following was the best solution found: a 272-DC public logistics network with a minimum average travel time of 8.999 hrs, as shown in Fig 5.1.

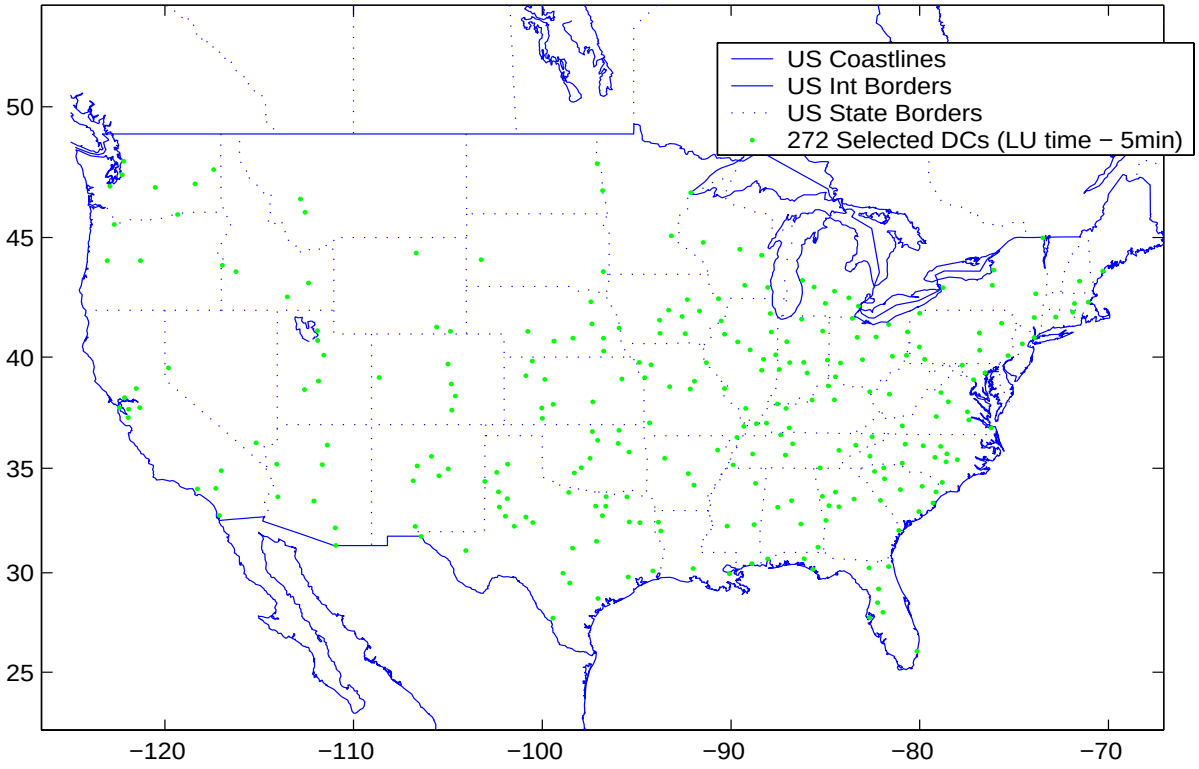


Fig 5.1 272 DCs selected in public logistics network with avg. travel time of 8.999 hrs

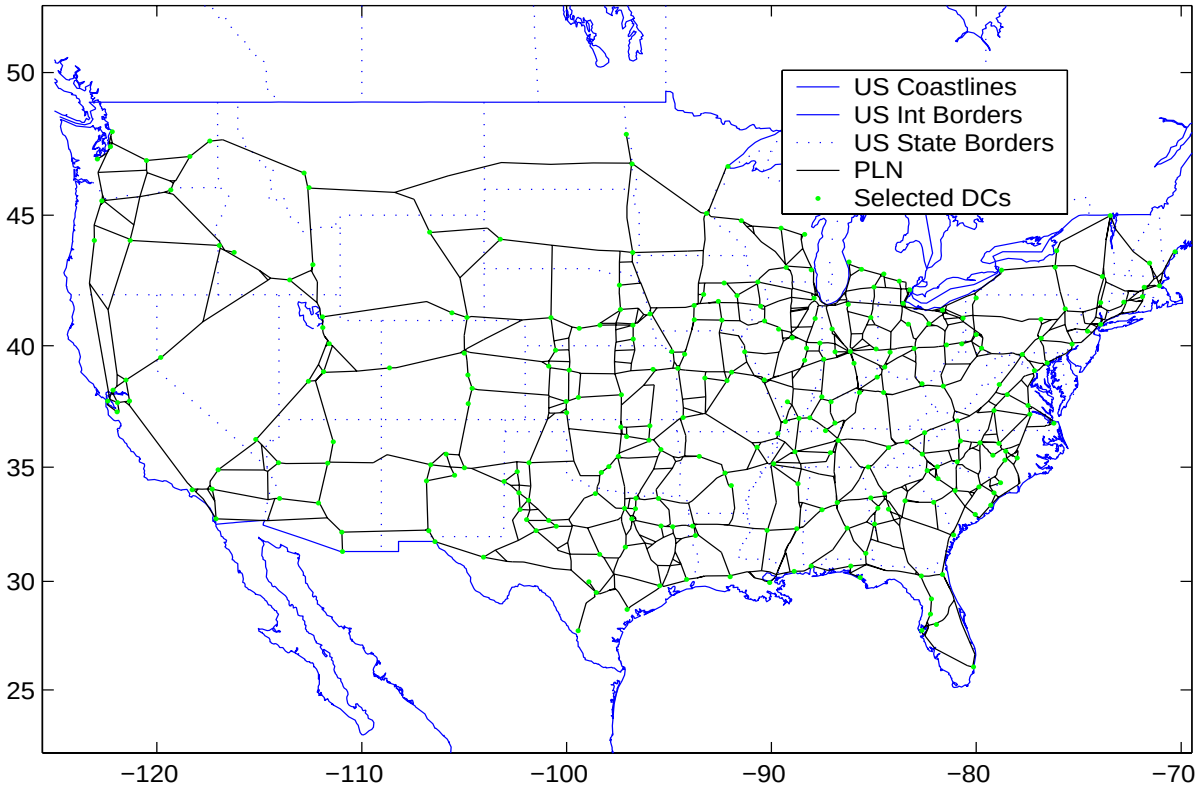


Fig 5.2 PLN of 272 selected DCs

Fig 5.3 shows the average travel time of the best solution found for different number of selected DCs in the PLN. It can be observed that the average travel time of the best solution found decreases as the number of DCs increases up to a point and then it starts increasing. The optimal solution is between 260 to 280 DCs. The initial decrease in average travel time associated with the increase in the number of DCs is due to the decrease in travel time of the trucks distance between the DCs. When the number of DCs reaches an optimal value, a further increase in the number of DCs increases the average travel time because then a package has to load and unload at a greater number of DCs, with the increase in loading/unloading time exceeding the decrease in travel time.

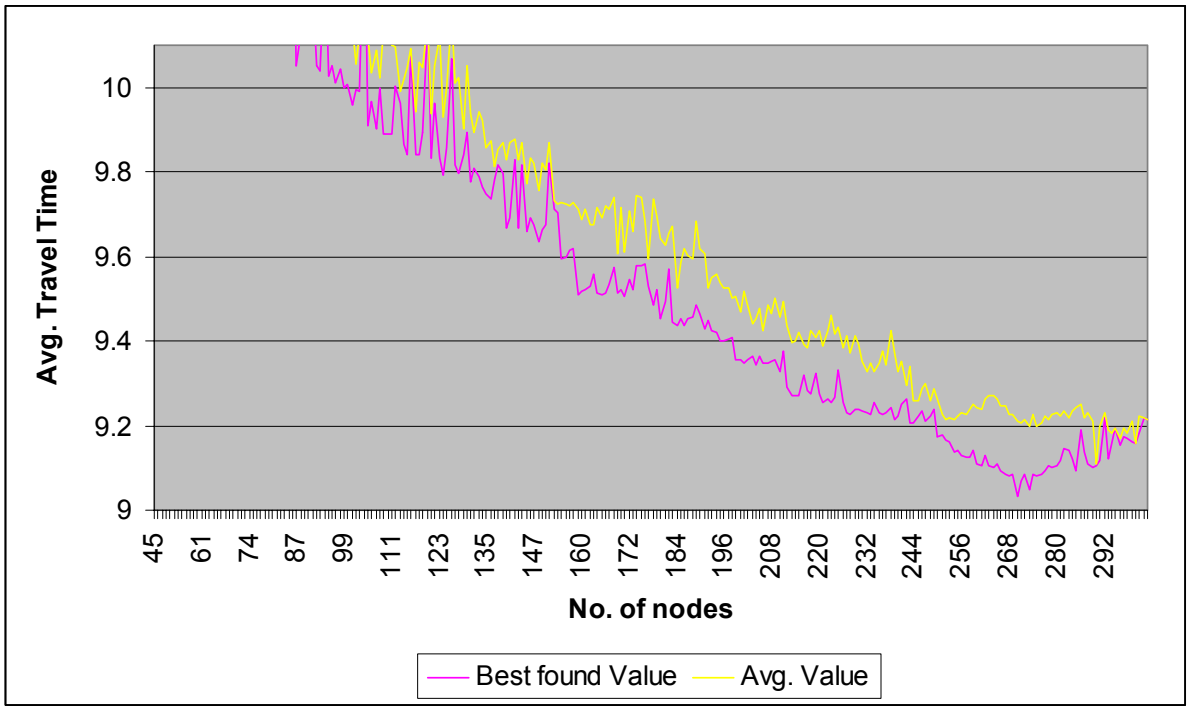


Fig 5.3 The best and the average of the Average Travel Time vs. Number of nodes in the network L/U time of 5 min.

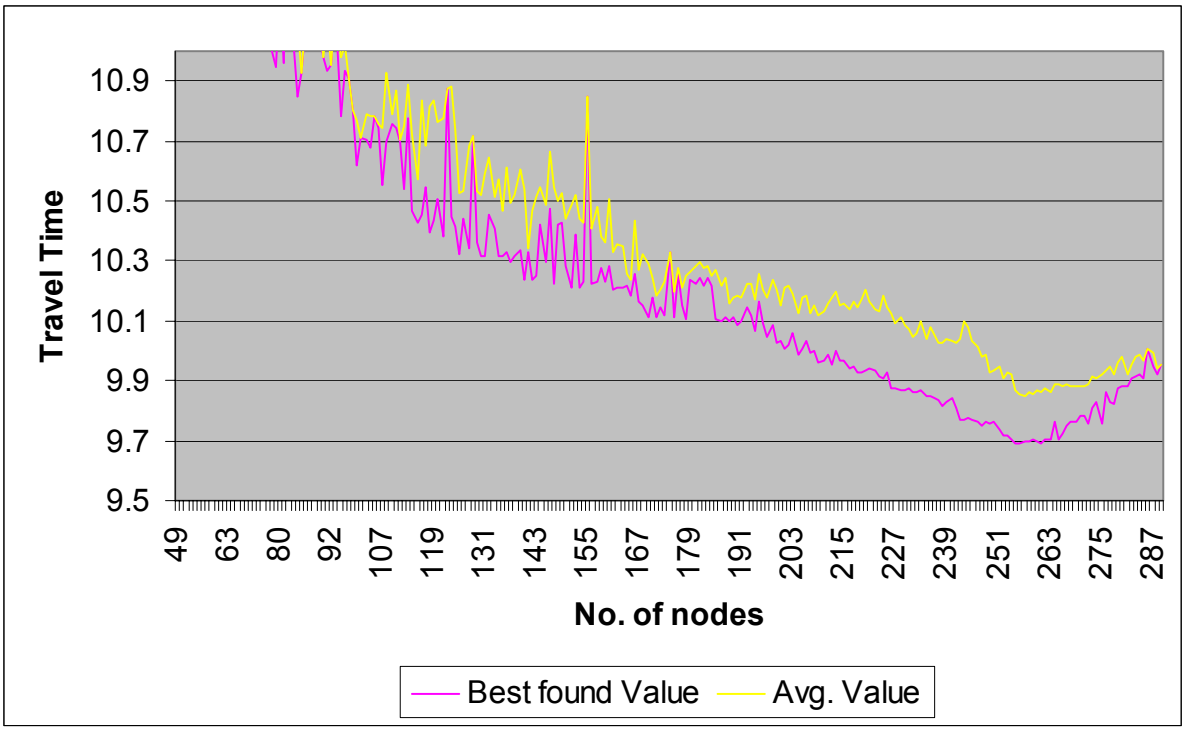


Fig 5.4 The best and the average of the Average Travel Time vs. Number of nodes in the network L/U time of 10 min.

Similar to the GA run when the loading and unloading time was 5 minutes, the GA was run for one complete day with a loading and unloading time of 10 minutes. Fig 5.4 shows the average travel time of the best solution found for different number of selected DCs in the PLN. As with the 5 minute GA run, the average travel time decreases as the number of DCs is increased but after it passes a point then it starts increasing.

The detailed results and the trace of the simulation runs can be seen in the Appendix 1B.

6. Conclusion and Future Work

6.1 Conclusion

The best average transport time of packages, when loading and unloading time is 5 min, is 8.999 hrs; and 272 DCs are selected in the network. Similarly, as expected, when loading and unloading time is 10 min, average transport time is longer. Number of DCs selected in the minimum average travel time network is also explainable since greater the loading and unloading time, then lesser the number of DCs a package should visit on its way to its final destination.

Different heuristics to design an underlying road network and the public logistics network has been proposed in this research and the genetic algorithm is used to calculate the network with minimum average travel time of packages. Method III seems more promising as it is giving the best road network, which can be seen from the results of the GA simulations.

6.2 Future Work

- Further extension of this work would be the possibility of applying this PLN approach to a local area covered by a DC for each and every DC selected in the entire area. The local demand generated in the local area will be transported just like the PLN described here and if this package needs to go out of this local area then it will go to the DC supporting the local area and from there it will be shipped to its destination DC.

- There is a possibility of looking at an approach to generate the initial underlying road network in which each and every I-I, I-U, U-U nodes or intersections (I-Interstates, U-U.S. highways) are studied and the demand is calculated at each point and the preference of selecting DCs at a node is in the following order

I – I > I - U > U - U

- There are some limitations on the values of parameters used in this research like the waiting time and the loading and unloading time. Actual DC has to be constructed to get the exact value of the above-mentioned time and the results are sensitive to these parameters.

- In addition, there are limitations on the data used as the exact knowledge of the speeds at each and every road in the network is not available since it keeps on changing and depends on the local conditions. For example, it is assumed that the speed of U.S. highways in rural areas is 55 miles per hour but there are many places in Nevada, Arizona where the speeds of these highways are 70 miles per hour. Therefore, we are lacking the exact data and the final solution is contingent upon all these parameters.

Bibliography

[1] Barrodale Computing Service Ltd., Automated Roading Tour, http://www.barrodale.com/roadingTour/index.html

[2] Genetic Algorithm Optimization Toolbox, http://www.ie.ncsu.edu/mirage/GAToolBox/gaot/

[3] J. B. Tenenbaum, V. de Silva, J. C. Langford (2000). "A global geometric framework for nonlinear dimensionality reduction," *Science* 290 (5500): 2319-2323, 22 December 2000.

[4] Jiang, D., Du, W., and Chen, X., "GA based location models for physical distribution centers" in *Proceedings of the 1997 IEEE International Conference on Intelligent Processing Systems*, Oct 28 – 31, Beijing, China, p. 553-557.

[5] Kay, M.G. and Parlikad, A.N., "Material Flow Analysis of Public Logistics Networks," in *Progress in Material Handling Research: 2002*, R. Meller et al., Eds., Charlotte, NC: The Material Handling Institute, 2002, pp. 205.

[5] Matlog: Logistics Engineering MATLAB Toolbox, http://www.ie.ncsu.edu/kay/matlog/index.htm

[7] U.S. Census Bureau. Transportation-Commodity Flow Survey, *1997 Commodity Flow Survey*, Washington D.C. Issued Dec 1999.

[8] United Parcel Service Inc., "2000 Annual Report." Securities and Exchange Commission. Washington. D.C.

APPENDIX 1A

MATLAB codes for running the simulation.

1. [*UnderRoadNetworkv3.m*] MATLAB code for generating underlying road network by method III.

```
function [fxy,fijd,ijd,xy,IJDio,XYio]= UnderRoadNetworkv3
% Finds the Underlying Road Network primarily composed of all US Interstates
% and the part of US highways.
%
%   [fxy,fijd] = UnderRoadNetwork(TMP,NoNodes,ArcLength)
%       fxy       = Final Node list
%       fijd      = Final Arc list
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% o=usrdlink;
% IJDspeed=SpeedAdj(o.IJD(:,3)>0,o.IJD);
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Extracting the Interstates and US highways from the whole US Roadways
% % Network excluding Canada, Mexico and Puerto Rico.
%
% [XY,IJD,isXY,isIJD] = subgraph(usrdnode('XY'), usrdnode('NodeFIPS')~= 72 &
    usrdnode('NodeFIPS')~= 88 & usrdnode('NodeFIPS')~= 91,...
% IJDspeed,usrdlink('Type')=='U' | usrdlink('Type')=='I');
%
% % Thin the US highways and Interstate Road Network.
% [tIJD,idxIJD] = thin(IJD);
% [stXY,stIJD,isstXY,isstIJD] = subgraph(XY,[],tIJD);
% [stXY,stIJD]=RemoveUnconnNodes(stXY,stIJD);
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%Above commented code is replaced by the saved network in ConnNetwork.mat
load ConnNetwork2
```

```
MinCityPopLim=10000;
s=uscity10k;
k=s.Pop>MinCityPopLim;
city20kp=s.XY(k,:);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
idx=argmin(dists(stXY,city20kp(1:end/2,:),'mi'));    % Indexes of nodes to which uszip5 is assigned.:
idx2=argmin(dists(stXY,city20kp(end/2+1:end,:),'mi'));    % Indexes of nodes to which uszip5 is assigned.:
citys=unique([idx,idx2]);
cityXY=stXY(citys,:);  % Nodes closest to the cities.
T=delaunay(cityXY(:,1),cityXY(:,2)); %Delaunay triangulation of selected nodes.
IJ=tri2list(T);
A=list2adj(stIJD);
padd=[];
p=size(IJ,1)
for i=1:p
i
[d2,p2]=dijk(A,citys(IJ(i,1)),citys(-IJ(i,2))); %shortest routes between the cities.
padd=[padd p2]; % All the nodes are collected.
end
finalarcs = unique(padd);
if ~isempty(padd)
[XYfinal,IJDfinal,isXYfinal,isIJDfinal] = subgraph(stXY,idx2is(padd,size(stXY,1)),stIJD);
end
[tIJC,idxIJC] = thin(IJDfinal);
[XY,IJD,isXY,isIJD] = subgraph(XYfinal, [],tIJC);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n=size(XY,1)   % No. of nodes in the network.
size(IJD)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% Following commented code removes the nodes with less than some
%%%%% nodevol from the above generated road network. [Works only with pop
%%%%% size greater than 20k, For 10k it need to be changed.
```

```
% DC=PopAndPos(XY);
%
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% % Find the Order Based proximity factor using 'proxfac' function.
% w1=DC.Pop/sum(DC.Pop);  % Pct. of total population.
% distance=dists(DC.XY,DC.XY,'mi');   % Distance between all the nodes.
% w=proxfac(distance,w1,0);  % Using the Proxfac function to find the order based proximity factor.
%
% clear distance
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tspeed    = 60;     % Truck Speed[60 mi/h]
% Demand    = 10434000; %%%Total demand for the whole country .........still need to subtract the demand of HI, AL
% LUtime    = 10/60;  % L/U time [10 min]
% IJDmod=[IJD(:,[1 2]),IJD(:,[3])+2*LUtime ]; % Adding Loading/Unloading time to the arcs.
% A=list2adj(IJDmod);
% e=size(A,1)
% for i=1:e
%     i
%     [d(i,1:e),p0]=dijk(A,i,1:e);
% end
% clear d
% [d0,p0]=dijk(A);    % Shortest Distance and Arcs between the nodes.
%  p=pred2path(p0);
% % Calculates the total DC-to-DC demand that is transported over the network.
% wnew=w*Demand;
% vol=zeros(n,n);
% for i=1:n
%     for j=1:n
%         if (i~=j)
%             vol([p{i,j}(1:end-1)],[p{i,j}(2:end)])=vol([p{i,j}(1:end-1)],[p{i,j}(2:end)])+wnew(i,j)*eye(size(p{i,j},2)-1);
```

```
%        end
%    end
% end
% clear p
% nodevol=[];
% for i=1:n
%    nodevol(i)=sum(vol(i,:))+sum(vol(:,i));
% end
%%%%%%%%%%%%%%%%5
%save 100000citydata IJD XY cityXY nodevol
%save 10000Ncitydata IJD XY cityXY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%plot(1:prod(size(nodevol)),sort(nodevol))
        [IJC11,IJC12,IJC22] = addconnector(cityXY,XY,IJD);

        IJCr = IJC12(IJC12(:,3)>0,:);
        IJC12n= [IJCr(:,1) IJCr(:,2) IJCr(:,3)/35]; % 35mi/h is the average speed to reach the city centre to the closest node.
        IJD=[IJC12n;IJC22];
        XY=[cityXY;XY];
        [XY,IJD]=subgraph(XY,[~idx2is(IJC12(IJC12(:,3)==0,1),size(cityXY,1))',(XY(size(cityXY,1)+1:end,1)>-1000)'],IJD);
        % Removing those nodes in the network which are on the network.
        makemap(XY);
        pplot(IJD,XY,'k-','Tag','Thinned Road Network')
        %pplot(IJDio,XYio,'r-','Tag','US Interstate Roads')
        pplot(cityXY,'g.')
        %selnodes=nodevol>0.02*max(nodevol);
        % selnodes=[ones(1,size(cityXY,1))>0,nodevol>0.05];
        % [XYr,IJDr,isXYr,isIJDr] = subgraph(XY,selnodes,IJD);
        %
        % [IJC11,IJC12,IJC22] = addconnector(cityXY,XYr,IJDr);
        % IJDr=[IJC12;IJC22];
        % XYr=[cityXY;XYr];
```

```
118.    %
119.    % makemap(XY)
120.    % pplot(IJDr,XYr,'k-','Tag','Thinned Road Network')
121.    % %pplot(IJDio,XYio,'r-','Tag','US Interstate Roads')
122.    % pplot(cityXY,'g.')
```

2. [*RemoveUnconnNodes.m*] MATLAB function for removing unconnected nodes

```
1. function [xy,ijd]=RemoveUnconnNodes(stxy,stijd)
2. %
3.   % Removes the nodes which are not connected to the whole network.
4.   %   [xy,ijd] = RemoveUnconnNodes(stxy,stijd)
5.   %       stxy  = Input nodes.
6.   %       stijd = Input Arcs.
7.   %       xy    = Output Nodes
8.   %       ijd   = Output Arcs.
9.
10. DI=dijkstra(sparse(list2adj(stijd)),1:size(stxy));
11. [I,J] = find(DI==Inf);
12. Rnodes=I(find(J==1));  % Any no. can be taken in place of (1). It should be less than the the
    no. of nodes.
13. [xy,ijd] = subgraph(stxy,~idx2is(Rnodes,size(stxy,1)),stijd);
```

3. [*AverageTravelTime.m*] MATLAB function for generating PLN and calling PLN function to calculate average travel time

```
1.  function [sol,avgTtime] = AverageTravelTime(sol,tXY,tIJD,WT,d0)
2. %
3. %
4.   % AverageTravelTime gives the average transport time on a network.
5.   %
6.   % [sol,avgTtime] = AverageTravelTime(sol,tXY,tIJD,WT,d0)
7.   %   sol  = Binary representation of Selected nodes(DC's) with one more
8.   %          dimension at the end.
9.   %   tXY  = XY cordinates of all nodes (DCs).
10.  %   tIJD = Arc list of the network.
11.  %   WT   = Population assigned to each node (DC in the network).
12.  %   d0   = Shortest distance matrix between all the nodes in the network (DC's).

13.  %function [sol,avgTtime] = AverageTravelTime(sol,options)

14.  % Loading all the input data for running GA because not able to figure
15.  % out a way to pass all this while running GA.

16.  load Data20k
17.  tXY=XY;    %xyi is the cordinates list
18.  tIJD=IJD;  % ijdi is the arc list
19.  WT=c;       % c is population assignment.

20.  global d0
21.  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % Parameters
22.  threshA=0.85;   % Edge is added to the new reduced network if its less than 85%
23.  % of the shortest actual arc distance.
24.  threshR=0.1;    % Edge is deleted from the new reduced network if it saves less
25.  % than 10% as compared to other two sides of the triangle.
```

```
% Other Parameters at the bottom of this function.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Selecting the first (n-1) element of the input binary vector which
% actually stores the information of the selected nodes (DC's) in the
% network for finding out the average travel time in the network.
x=sol(1:end-1);

b=find(x);  %Find the actual indexes of the selected nodes.
%disp(prod(size(b)));    % Display the no. of nodes Selected.
nXY=tXY(b,:);   %It stores the actual XY cordinates of the selected nodes.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reducing the entire road network to a smaller one with all selected nodes
% as the only nodes in the network using the delaunay triangulation.
% ............see details in report.
T=delaunay(nXY(:,1),nXY(:,2)); %Delaunay triangulation of selected nodes.
IJ=tri2list(T);
A=list2adj(tIJD);
nIJD=[];
xxIJD=[];
for i=1:size(IJ,1)
p=d0(b(IJ(i,1)),b(-IJ(i,2)));   % Shortest arc length of the sides of the triangle.
nIJD=[nIJD;b(IJ(i,1)),-b(-IJ(i,2)),p];  % This is IJD matrix of new network with indexes of the original network.
xxIJD=[xxIJD;IJ(i,1),IJ(i,2),p];    % This is IJD matrix of new network with reduced new indexes.
end
% Finds the shortest distance between all the nodes of the reduced new
% network to be used further in this algorithm.
C=list2adj(xxIJD);
c=dijkstra(sparse(C),1:size(C,1));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Adding some new arcs in the new reduced network obtained by delaunay
% triangulation. If the direct distance, between the opposite nodes of the
```

```
% adjacent triangles, in actual network is less than some threshold of
% their shortest distance in the new network then that arc is added to this
% new network.
xIJD = Connector(T,b,d0,threshA,c); % Calls the connector function which returns
% the new arcs that need to be added to this network.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Removing some Arcs from the new reduced network. For. E.g. if one side of
% a triangle is just saving less than threshold of the distance between
% them by other two sides of the triangle then remove this arc.
rIJD = RemoveArcs(T, b, threshR, c);    % Returns the IJD matrix of the arcs to be
% removed from the new network.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plot the nodes(DC locations) and arcs included in the final arc list.
%PlotRoadNetwork(tXY,xIJD,nXY,IJ,rIJD,nIJD)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Find the population assigned to each node in the final network and the
% average weighted distance of each node to the uszip5 locations assigned to
% that node.
DC = PopAndPos(nXY);    % Return a structure with XY cordinates,Pop population and Dist
    avg. weighted distance
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Getting the final IJD matrix to determine Avg. travel time in the PLN
%
% fIJD is obtained by removing the rIJD arcs from the original network
if ~isempty(rIJD)
fIJD=adj2list(list2adj(nIJD,size(tXY,1))-list2adj(rIJD,size(tXY,1)));
else
fIJD=nIJD;
end
% Filter fIJD to remove the arcs which occured twice in rIJD matrix and
% thus removing the arcs of fIJD with negative arc length.
fIJD=fIJD(find(fIJD(:,3)>0.01),:);
% Rewrite fIJD in new indexes form. IJD2 is the final network after
% removing the undesired arcs.
```

```
88. IJD2=zeros(size(fIJD,1),3);
89. IJD2(:,3)=fIJD(:,3);
90. for i=1:size(fIJD,1)
91. IJD2(i,1)=find(b==fIJD(i,1));
92. IJD2(i,2)=-find(b==abs(fIJD(i,2)));
93. end
94. % IJD1 are the new arcs added to the network.
95. IJD1=xIJD;
96. % Final IJD matrix which will be used to calculate the Avg Transport Time in PLN.
97. IJD=[IJD1;IJD2];
98. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
99. % Calculates the Average Transport Time using the following Parameters.
100.        %Tspeed    = 60;     % Truck Speed[60 mi/h]
101.        Tload      = 60;     % Truck Load [60 Packages/Truck]
102.        Lfactor   = 0.8;    % Load Factor [80% of Full-Truck]
103.        Demand    = 10434000; %%%Total demand for the whole country .........still need to
    subtract the demand of HI, AL
104.        Dfactor   = 1;    % Demand Factor [0.5]
105.        LUtime    = 10/60;  % L/U time [10 min]
106.        prox      = 6.55625637325098;   % Proximity Factor
107.        TspeedLocal = 45;   % Speed of Local Transport from U.S.ZIP5 to a DC.
108.        avgTtime = -PLNnew(DC, IJD, Tload, Lfactor, Demand, Dfactor, LUtime, prox,
    TspeedLocal); % -ve to min the AVg T Time
109.        disp([prod(size(b)) avgTtime ]);    % Display the no. of nodes and Avg. Travel Time
    of PLN
```

4. [*PLN.m*] MATLAB function for calculating average travel time of PLN.

```
1. function [avgTtime] = PLN(DC, IJD, Tload, Lfactor, Demand, Dfactor, LUtime, prox,
TspeedLocal)
2. %
3. %
4.  %
5.   % PLN Average Transport Time estimates using Public Logistics Network
6.   %
7.   % [avgTtime] = PLN(DC, IJD, Tspeed, Tload, Lfactor, Demand, Dfactor, LUtime, prox)
8.   %
9.   % DC        = It contains the population,location and average weighted
10. %             distance(from all U.S.ZIP5 locations covered by it) of the
11. %             node(DC's).
12. % IJD       = Public Logistics Network.
13. % Tspeed    = Truck Speed
14. % Tload     = Max. Load Capacity of truck
15. % Lfactor   = Average load factor of truck
16. % Demand    = UPS daily demand in region
17. % Dfactor   = Percentage of UPS demand [ 50% = 0.5 ]
18. % LUtime    = Loading/Unloading time [ 10 min = 10/60 hrs]
19. % prox      = Proximity factor
20. % TspeedLocal   = Speed of Trucks transferring stuff from uszip5 locations
21. %                to the DC.
22. n=size(DC.Pop,2);   % No. of nodes in the network.
23. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
24. % Find the Order Based proximity factor using 'proxfac' function.
25. w1=DC.Pop/sum(DC.Pop);  % Pct. of total population.
26. distance=dists(DC.XY,DC.XY,'mi');   % Distance between all the nodes.
27. w=proxfac(distance,w1);  % Using the Proxfac function to find the order based proximity
    factor.
28. %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
29. %IJDmod=[IJD(:,[1 2]),IJD(:,[3])+2*LUtime*Tspeed ]; % Adding Loading/Unloading time
    to the arcs.
```

```
IJDmod=[IJD(:,[1 2]),IJD(:,[3])+2*LUtime ]; % Adding Loading/Unloading time to the arcs.
A=list2adj(IJDmod);
[d0,p0]=dijk(A);    % Shortest Distance and Arcs between the nodes. [d0 is already time]
p=pred2path(p0);
dij=(DC.Dist)'*ones(1,n);
dij=dij+dij';   % By adding its transpose, we are actually adding the average weighted
    distances of starting and end node.
% Travel Time is sum of time taken from U.S.ZIP5 to a DC, vice-versa and from
% Starting DC to an end DC. Local demand within a DC is added automatically
% in this calculation because diag(d0)==0. So there is no transfer of local
% package from one DC to another. They will just come to a DC and then get
% redistributed again to the other U.S.zip5's in the same DC.
Ttime=d0 + dij*1.2/TspeedLocal;  % dij is great circle distances. So adding
%                                   20%(estimate) to make it a road distance.
% Calculates the total DC-to-DC demand that is transported over the network.
wnew=w*Demand*Dfactor/(Tload*Lfactor);
vol=zeros(n,n);
for i=1:n
for j=1:n
if (i~=j)
vol([p{i,j}(1:end-1)],[p{i,j}(2:end)])=vol([p{i,j}(1:end-
    1)],[p{i,j}(2:end)])+wnew(i,j)*eye(size(p{i,j},2)-1);
end
end
end
wtime=24./(vol+0.000001);% Truck waiting time in a link(arc).
% Calculates the total time package needs to wait for the trucks from starting to the end.
WFTT=zeros(n,n);
for i=1:n
for j=1:n
if (i~=j)
WFTT(i,j)= sum(diag(wtime([p{i,j}(1:end-1)],[p{i,j}(2:end)]))); % Wait for Truck time
    between any 2 DC's.
end
```

```
61. end
62. end
63. Totaltime=WFTT+Ttime;   % Total time = Travel Time [Within DC's(only at start and end)
    and from one DC to another DC + L/U time + Wait for Truck Time
64. avgTtime=sum(sum(Totaltime.*w));    % Average Transport Time
```

5. [*InitWt.m*] MATLAB function for calculating U.S. population distribution for each node in the network

```
1. function WT = InitWt(xy)
2. % InitWT Find the U.S. population distribution for each node in the network
3. %   xy  =   XY cordinates of the nodes in the network
4. %
5. %
6.   %   WT  =   Percentage of Total population associated with each node.
7.   [XY,Pop]=uszip5('XY','Pop',~strcmp('HI',uszip5('ST'))&~strcmp('AK',uszip5('ST'))&~strcm
     p('PR',uszip5('ST')));
8.   for i=1:size(XY,1)
9.   idx(i)=argmin(dists(XY(i,:),xy,'mi'));
10. end
11. for j=1:size(xy)
12. WT(j)=sum(Pop(find(idx==j)));
13. End
```

6. [*RemoveArcs.m*] MATLAB function to remove the arcs from the network obtained by delaunay triangulation of the selected nodes.

```
1.  function [rIJD] = RemoveArcs(T, b, thresh,c)
2.  % RemoveArcs Find the arcs to be removed from a new network obtained by a
3.  % delaunay triangulation of the selcted nodes from the original network.
4.  %
5.  % [rIJD] = RemoveArcs(b, T, h,c)
6.  %   T   =   Delaunay triangulation of the original network.
7.  %   b   =   Actual indexes of selected nodes from the original network.
8.  %   thresh   =   Parameter (threshold to remove an arc in the network)[thresh = 0.1]
9.  %   c   =   The shortest distance matrix between all the nodes of the reduced new
10. %           network obtained by delaunay triangulation.
11. %   rIJD    =   IJD matrix of the arcs to be deleted from the network.
12. rIJD=[];
13. % Check every triangle for potential side(arc of a network) to be removed.
14. for i=1:size(T,1)
15. s1=c(T(i,1),T(i,2));    % Shortest length of three sides of a tringle.
16. s2=c(T(i,2),T(i,3));
17. s3=c(T(i,3),T(i,1));
18. % If the largest side of the triangle is less than the 100 + threshold
19. % percentage of the sum of the other two sides than remove the largest
20. % side.
21. [Y,I] = max([s1, s2, s3]);
22. if ((s1+s2+s3) - 2*Y) < thresh*Y
23. rIJD=[rIJD;b(T(i,I)),-b(T(i,(I+1-3*((I+1)==4)))),Y ];
24. end
25. end
```

7. [*Connector.m*] MATLAB function to add new arcs in the network obtained by Delaunay triangulation of the selected nodes.

```
function [xIJD] = Connector(T,b,d0,thresh,c)
% Connector Find the new arcs to be added to a reduced network obtained by
% delaunay triangulation of the orignial network.
%
% [xIJD] = Connector(T,b,d0,thresh,c)
%  T  =  Delaunay triangulation of the original network.
%  b  =  Actual indexes of selected nodes from the original network.
%  d0 =  Shor   test distance matrix between all the nodes in the network (DC's).
%  thresh  =  Parameter (threshold to add an arc in the network)[thresh = 0.85]
%  c  =  The shortest distance matrix between all the nodes of the reduced new
%        network obtained by delaunay triangulation.
%  xIJD   =  IJD matrix of the arcs to be added in the network.
% It finds neighbors of a triangle.
N=trineighbors(1:size(T,1),T);
xIJD=[];
for i=1:size(N,1)
for j=1:size(N,2)
if(N(i,j)~=NaN & N(i,j)>i)
x=setxor(T(i,:),T(N(i,j),:));   % Returns the nodes that are not common in adjacent triangles
% or the opposite nodes of the adjacent triangles.
do=d0(b(x(1)),b(x(2))); % Shortest Distance in the original network.[do]
dt=c(x(1),x(2));    % Shortest Distance in the new reduced delaunay triangulated network.[dt]
if (do < thresh*dt)     % Threshold is 85% i.e. Arc is chosen only if it saves more than 15% of
distance.
xIJD=[xIJD;x(1),-x(2),do];   % Updating final IJD matrix of new arcs.
end
end
end
end
```

8. [*PopAndPos.m*] MATLAB function to calculate the position, population and avg. weighted distance of a node.

```
function [DC]=PopAndPos(nXY)
% PopAndPop Return a structure with Position, Population and average
% weighted distance of a node(XY cordinates of a DC).
%
%   DC = PopAndPos(nXY)
%   nXY =   XY cordinates of nodes(DC's)
%   DC  =   Structure with 3 fields(XY, Pop, Dist)
%       DC.XY   =   XY cordinates of nodes(DC's).
%       DC.Pop  =   Population assigned to each node (DC).
%       DC.Dist =   Avg. weighted distance between a node (DC) and all
%                   U.S.ZIP5 locations covered by it
% Extracting all XY cordinates and Population from U.S. 5-digit ZIP code data
% excluding Hawai, Alaska and Puerto Rico.
[zipXY,zipPop]=uszip5('XY','Pop',~strcmp('HI',uszip5('ST'))&~strcmp('AK',uszip5('ST'))&~
strcmp('PR',uszip5('ST')));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Find which uszip5 location is covered or assigned to which node(DC) based
% on the proximity of a uszip5 location to a node.
% For approximate and fast computation.
T = delaunay(nXY(:,1),nXY(:,2));  % Delaunay triangulation of input nodes
n = prod(size(nXY(:,1)));   % No. of nodes
S = sparse(T(:,[1 1 2 2 3 3]),T(:,[2 3 1 3 1 2]),1,n,n);
idx = dsearch(nXY(:,1),nXY(:,2),T,zipXY(:,1),zipXY(:,2)); % Indexes of nodes to which
uszip5 is assigned.
% For precise and time consuming computation.
% idx=argmin(dists(nXY,zipXY,'mi'));    % Indexes of nodes to which uszip5 is assigned.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:size(nXY,1)
indexes=find(idx==i);   % All U.S.ZIP5 locations covered by a node.
if ~isempty(indexes)
WT(i)=sum(zipPop(indexes));     % Total population covered by a node(DC).
```

```
30. sumidx2=dists(zipXY(indexes,:),nXY(i,:),'mi').*zipPop(indexes); % Vector of Distance X
    Population
31. Dist(i)=sum(sumidx2)/WT(i); % Avg. weighted distance between a node(DC) and all
    U.S.ZIP5 locations covered by it.
32. else
33. WT(i)=0;    % If no U.S.ZIP5 locations are covered by a node then its 0.
34. Dist(i)=Inf;    % If no U.S.ZIP5 locations are covered by a node then its Inf.
35. end
36. end
37. DC.XY=nXY;
38. DC.Pop=WT;
39. DC.Dist=Dist;
```

9. [*GenerateinitialSolGA.m*] MATLAB script to run GA simulation

```
% This script generate the initial population and provide all the parameters
needed to run the GA.
%
% This program loads 'TEMP' which has XY cordinates('xyi') and IJD
% matrix('ijdi') of the PLN network and the weight of each node ('c') in
% the network which is calculated by time consuming operation and the
% shortest distance between all the nodes in the network given by ('d0').
load data20k % Network data that need to be evaluated
tXY=XY;
tIJD=IJD;
WT=c;   % c = InitWt(XY);
%global d0
d0 = dijkstra(sparse(list2adj(IJD)),1:size(XY,1));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate initial population
n=size(tXY,1);
p=0.01
PopSize=10
rand('state',sum(100*clock)) %resets it to a different state each time.
m=sum(100*clock);
%b=ceil(rand(100,1)*size(tXY,1)); % Randomly selected 100 DCs for PLN in this network...
%b=wtrandperm(WT,100);
startPop=zeros(PopSize,n+1);
for i=1:PopSize
%Selection of DCs with probability using the population around them as
%the weight.
b{i}=find(rand(1,length(WT)) < p * WT/mean(WT));
startPop(i,1:n)=idx2is(b{i},n);
[zzz,startPop(i,n+1)]=AverageTravelTime(startPop(i,:));
[q1,w1]=AverageTravelTime(startPop(i,:),XY,IJD,c,d0);
end
```

```
%startPop=sparse(startPop); % It gives error with this sparse matrix.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Crossover Operators
xFns = 'simpleXover';
xOpts = [.4];
% Mutation Operators
mFns = 'binaryMutation';
mOpts = [0.002];
% Termination Operators
termFns = 'maxGenTerm';
termOps = [50]; % 200 Generations
% Selection Function
selectFn = 'roulette';
selectOps = [];
% Evaluation Function
evalFn = 'AverageTravelTime';
evalOps = [];
% Bounds on the variables
bounds=[zeros(n,1) ones(n,1)*2];
%bounds=[];
gaOpts=[1e-6 0 1];
%startPop = initializega(10,bounds,'triangle2',[],[1e-6 0]);
[x endPop bestPop trace]=ga(bounds,evalFn,evalOps,startPop,gaOpts,...
termFns,termOps,selectFn,selectOps,xFns,xOpts,mFns,mOpts);
```

10. [*PlotRoadNetwork.m*] MATLAB function to plot road networks.

```
function [] = PlotRoadNetwork(tXY,xIJD,nXY,IJ,rIJD,nIJD)
% PlotRoadNetwork Plot the Underlying network
makemap(tXY)
if ~isempty(xIJD)
        pplot(xIJD,nXY,'m-')    % Plot the additional arcs in magenta color.
        pplot(xIJD,num2cell(xIJD(:,3)),nXY) % Plot the arc distances in the network.
end
pplot(IJ,nXY,'g-')  % Plot the original reduced network obtained ny delaunay triangulation.
if ~isempty(rIJD)
        pplot(rIJD(:,[1 2]),tXY,'k-')   % Plot the arcs removed from the network in black color.
end
pplot(nIJD(:,[1 2]),num2cell(nIJD(:,3)),tXY)    % Plot the arc distances in the network.
```

11. [*proxfac.m*] MATLAB function to find order based proximity function.

```
function W = proxfac(D,w,p)
%PROXFAC Order-based proximity factor.
% W = proxfac(D,w,p)
%     D = n x n distance matrix
%     w = n-element marginal weight
%     p = proximity factor (p == 0 => no proximity adjustment)
%       = 6.55625637325098, default
%     W = n x n weight matrix, where, for p = 0, W(i,j) = w(i)*w(j)
%
% Default based on LS fit to 1997 State-to-State Commodity Flow data
% on the value of shipments (http://www.bts.gov/ntda/cfs/cfs97od.html)
% with 95% confidence interval [6.4139, 6.6986].
% Copyright (c) 1994-2003 by Michael G. Kay
% Matlog Version 7 25-Sep-2003
% Input Error Checking ************************************************
if nargin < 3, p = 6.55625637325098; end
% End (Input Error Checking) ******************************************
m = length(D);
W0 = w(:)*w(:)';
I = argsort(argsort(D,2),2);
R = ((1-p/m).^(I-1))/(mean((1-p/m).^((1:m)-1)));
W = R.*W0./sum(sum(R.*W0));
```

12. [*PrepInitSol.m*] MATLAB function to remove some nodes from the solution such that the distance between any two nodes is greater than given threshold value.

```
function [solnew] = PrepInitSol(sol,tXY,ThreshDistance)
% PrepInitSol removes the nodes such that the distance between any two
% nodes is greater than given threshold value.
%
% [solnew] = PrepInitSol(sol,tXY,ThreshDistance)
still=0;
while(~still)
x=sol(1:end-1);
b=find(x);  %Find the actual indexes of the selected nodes.
nXY=tXY(b,:);   %It stores the actual XY cordinates of the selected nodes.
[DC]=PopAndPos(nXY);
dt=dists(nXY,nXY,'mi');
[e,r]=sort(dt,2);
bin=[];
for i=1:size(nXY,1);
        if ~prod(size(find(bin==i)))>0
            if dt(i,r(i,2))<ThreshDistance;
                if (DC.Pop(i)>DC.Pop(r(i,2)))
                    bin=[bin,r(i,2)];
                else
                    bin=[bin,i];
                end
            end
        end
end
if ~isempty(bin)
        nodesnottokeep=setxor(b(bin),b);
        solnew=idx2is(nodesnottokeep,prod(size(sol)));
else
    solnew=sol;
    still=1;    end    sol=solnew; end solnew=sol;
```

APPENDIX 1B

Detailed results of the GA simulation

When loading and unloading time was 5 min and the initial population size was 10. The following table shows the trace of the GA output.

Table A.1 Trace of the GA simulation

Iteration No.	Best Solution found so far	Avg. Value of the population	Iteration No.	Best Solution found so far	Avg. Value of the population	Iteration No.	Best Solution found so far	Avg. Value of the population
1	11.419	12.227	46	9.668	9.815	91	9.399	9.526
2	11.167	11.779	47	9.666	9.784	92	9.389	9.467
3	11.167	11.500	48	9.661	9.809	93	9.354	9.495
4	10.923	11.292	49	9.650	9.774	94	9.354	9.421
5	10.721	11.084	50	9.650	9.787	95	9.354	9.401
6	10.721	11.023	51	9.645	9.764	96	9.354	9.403
7	10.692	10.901	52	9.636	9.706	97	9.349	9.407
8	10.560	10.753	53	9.604	9.776	98	9.349	9.429
9	10.313	10.636	54	9.604	9.738	99	9.343	9.428
10	10.285	10.506	55	9.597	9.724	100	9.343	9.427
11	10.285	10.549	56	9.511	9.765	101	9.343	9.439
12	10.285	10.507	57	9.511	9.724	102	9.343	9.449
13	10.285	10.472	58	9.511	9.660	103	9.343	9.465
14	10.185	10.424	59	9.511	9.643	104	9.343	9.460
15	10.147	10.348	60	9.511	9.631	105	9.343	9.468
16	10.052	10.295	61	9.511	9.627	106	9.343	9.485
17	10.052	10.200	62	9.511	9.669	107	9.343	9.493
18	10.039	10.129	63	9.511	9.658	108	9.343	9.484
19	10.026	10.109	64	9.511	9.617	109	9.343	9.465
20	9.958	10.080	65	9.511	9.641	110	9.343	9.455
21	9.958	10.084	66	9.511	9.627	111	9.343	9.461
22	9.902	9.990	67	9.510	9.585	112	9.343	9.475
23	9.888	10.004	68	9.510	9.553	113	9.343	9.446
24	9.888	10.010	69	9.510	9.574	114	9.343	9.465
25	9.888	10.020	70	9.510	9.581	115	9.343	9.470
26	9.888	10.023	71	9.485	9.579	116	9.343	9.427
27	9.888	10.085	72	9.485	9.574	117	9.331	9.395
28	9.888	10.138	73	9.455	9.560	118	9.299	9.362
29	9.865	10.102	74	9.455	9.552	119	9.299	9.363
30	9.865	10.079	75	9.447	9.522	120	9.299	9.360
31	9.840	9.998	76	9.436	9.509	121	9.299	9.366
32	9.833	9.917	77	9.436	9.504	122	9.299	9.361
33	9.792	9.916	78	9.435	9.515	123	9.299	9.348
34	9.792	9.914	79	9.435	9.522	124	9.286	9.348
35	9.792	9.902	80	9.427	9.512	125	9.269	9.334
36	9.792	9.897	81	9.427	9.497	126	9.269	9.346
37	9.782	9.881	82	9.421	9.466	127	9.269	9.323
38	9.782	9.866	83	9.421	9.475	128	9.260	9.326
39	9.782	9.829	84	9.421	9.488	129	9.256	9.336
40	9.777	9.828	85	9.402	9.500	130	9.256	9.345
41	9.767	9.812	86	9.402	9.493	131	9.239	9.343
42	9.749	9.811	87	9.402	9.505	132	9.239	9.334
43	9.749	9.828	88	9.402	9.510	133	9.239	9.342
44	9.734	9.828	89	9.402	9.505	134	9.231	9.311
45	9.693	9.823	90	9.399	9.530	135	9.231	9.323

Iteration No.	Best Solution found so far	Avg. Value of the population	Iteration No.	Best Solution found so far	Avg. Value of the population	Iteration No.	Best Solution found so far	Avg. Value of the population
136	9.231	9.314	181	9.167	9.228	226	9.164	9.243
137	9.231	9.322	182	9.167	9.218	227	9.164	9.248
138	9.231	9.316	183	9.167	9.219	228	9.164	9.235
139	9.225	9.299	184	9.167	9.218	229	9.164	9.267
140	9.225	9.284	185	9.167	9.227	230	9.164	9.260
141	9.225	9.272	186	9.167	9.209	231	9.155	9.239
142	9.225	9.269	187	9.167	9.213	232	9.137	9.238
143	9.225	9.247	188	9.167	9.222	233	9.137	9.278
144	9.225	9.260	189	9.167	9.238	234	9.137	9.265
145	9.225	9.283	190	9.167	9.250	235	9.137	9.258
146	9.213	9.289	191	9.167	9.243	236	9.137	9.224
147	9.208	9.294	192	9.167	9.237	237	9.137	9.246
148	9.208	9.279	193	9.167	9.248	238	9.137	9.232
149	9.208	9.267	194	9.167	9.231	239	9.127	9.248
150	9.208	9.268	195	9.167	9.224	240	9.127	9.229
151	9.208	9.276	196	9.167	9.203	241	9.127	9.227
152	9.207	9.266	197	9.167	9.231	242	9.127	9.249
153	9.207	9.271	198	9.167	9.245	243	9.127	9.300
154	9.206	9.269	199	9.167	9.271	244	9.127	9.286
155	9.206	9.273	200	9.167	9.283	245	9.127	9.254
156	9.206	9.251	201	9.167	9.284	246	9.124	9.245
157	9.187	9.275	202	9.167	9.246	247	9.124	9.276
158	9.187	9.314	203	9.167	9.253	248	9.124	9.286
159	9.187	9.304	204	9.167	9.254	249	9.124	9.295
160	9.187	9.316	205	9.167	9.246	250	9.124	9.257
161	9.187	9.335	206	9.167	9.225	251	9.124	9.234
162	9.187	9.363	207	9.167	9.226	252	9.124	9.260
163	9.187	9.353	208	9.167	9.223	253	9.124	9.245
164	9.187	9.389	209	9.164	9.231	254	9.108	9.187
165	9.187	9.357	210	9.164	9.229	255	9.107	9.177
166	9.187	9.308	211	9.164	9.210	256	9.104	9.162
167	9.185	9.323	212	9.164	9.210	257	9.104	9.160
168	9.185	9.312	213	9.164	9.218	258	9.104	9.154
169	9.185	9.322	214	9.164	9.215	259	9.104	9.172
170	9.185	9.308	215	9.164	9.213	260	9.104	9.180
171	9.185	9.339	216	9.164	9.199	261	9.104	9.191
172	9.172	9.282	217	9.164	9.208	262	9.104	9.201
173	9.172	9.260	218	9.164	9.214	263	9.104	9.212
174	9.172	9.236	219	9.164	9.225	264	9.104	9.216
175	9.172	9.228	220	9.164	9.218	265	9.104	9.242
176	9.172	9.216	221	9.164	9.230	266	9.104	9.229
177	9.167	9.203	222	9.164	9.217	267	9.104	9.222
178	9.167	9.219	223	9.164	9.226	268	9.103	9.204
179	9.167	9.235	224	9.164	9.237	269	9.103	9.204
180	9.167	9.229	225	9.164	9.243	270	9.103	9.177

Iteration No.	Best Solution found so far	Avg. Value of the population	Iteration No.	Best Solution found so far	Avg. Value of the population	Iteration No.	Best Solution found so far	Avg. Value of the population
271	9.103	9.195	316	9.089	9.155	361	9.026	9.081
272	9.103	9.163	317	9.089	9.152	362	9.026	9.090
273	9.103	9.161	318	9.089	9.140	363	9.025	9.075
274	9.103	9.166	319	9.089	9.147	364	9.025	9.076
275	9.103	9.167	320	9.089	9.157	365	9.025	9.075
276	9.103	9.192	321	9.089	9.173	366	9.025	9.090
277	9.103	9.186	322	9.083	9.154	367	9.022	9.089
278	9.103	9.177	323	9.083	9.155	368	9.020	9.081
279	9.103	9.150	324	9.083	9.164	369	9.020	9.082
280	9.103	9.152	325	9.083	9.174	370	9.019	9.050
281	9.103	9.171	326	9.083	9.159	371	9.016	9.051
282	9.103	9.163	327	9.082	9.139	372	9.016	9.046
283	9.103	9.162	328	9.082	9.137	373	9.016	9.055
284	9.103	9.171	329	9.082	9.153	374	9.016	9.062
285	9.103	9.158	330	9.082	9.160	375	9.016	9.082
286	9.102	9.159	331	9.082	9.160	376	9.016	9.075
287	9.102	9.156	332	9.082	9.134	377	9.016	9.074
288	9.102	9.146	333	9.082	9.154	378	9.016	9.078
289	9.102	9.158	334	9.082	9.136	379	9.016	9.088
290	9.094	9.137	335	9.072	9.129	380	9.016	9.099
291	9.094	9.122	336	9.072	9.124	381	9.016	9.119
292	9.094	9.128	337	9.072	9.132	382	9.016	9.122
293	9.094	9.146	338	9.072	9.136	383	9.014	9.131
294	9.094	9.192	339	9.069	9.136	384	9.014	9.127
295	9.094	9.205	340	9.068	9.132	385	9.014	9.124
296	9.094	9.191	341	9.068	9.117	386	9.013	9.103
297	9.092	9.184	342	9.068	9.144	387	9.013	9.117
298	9.092	9.212	343	9.068	9.146	388	9.013	9.126
299	9.092	9.214	344	9.049	9.120	389	9.013	9.107
300	9.092	9.243	345	9.049	9.112	390	9.013	9.134
301	9.092	9.241	346	9.031	9.109	391	9.013	9.115
302	9.089	9.236	347	9.031	9.119	392	9.013	9.115
303	9.089	9.220	348	9.031	9.109	393	9.001	9.132
304	9.089	9.206	349	9.031	9.094	394	9.001	9.144
305	9.089	9.180	350	9.031	9.108	395	9.001	9.138
306	9.089	9.153	351	9.031	9.068	396	9.001	9.107
307	9.089	9.161	352	9.031	9.068	397	9.001	9.098
308	9.089	9.169	353	9.031	9.072	398	9.001	9.107
309	9.089	9.162	354	9.031	9.079	399	9.001	9.128
310	9.089	9.130	355	9.026	9.080	400	8.999	9.137
311	9.089	9.136	356	9.026	9.092			
312	9.089	9.129	357	9.026	9.062			
313	9.089	9.139	358	9.026	9.069			
314	9.089	9.143	359	9.026	9.067			
315	9.089	9.153	360	9.026	9.074			

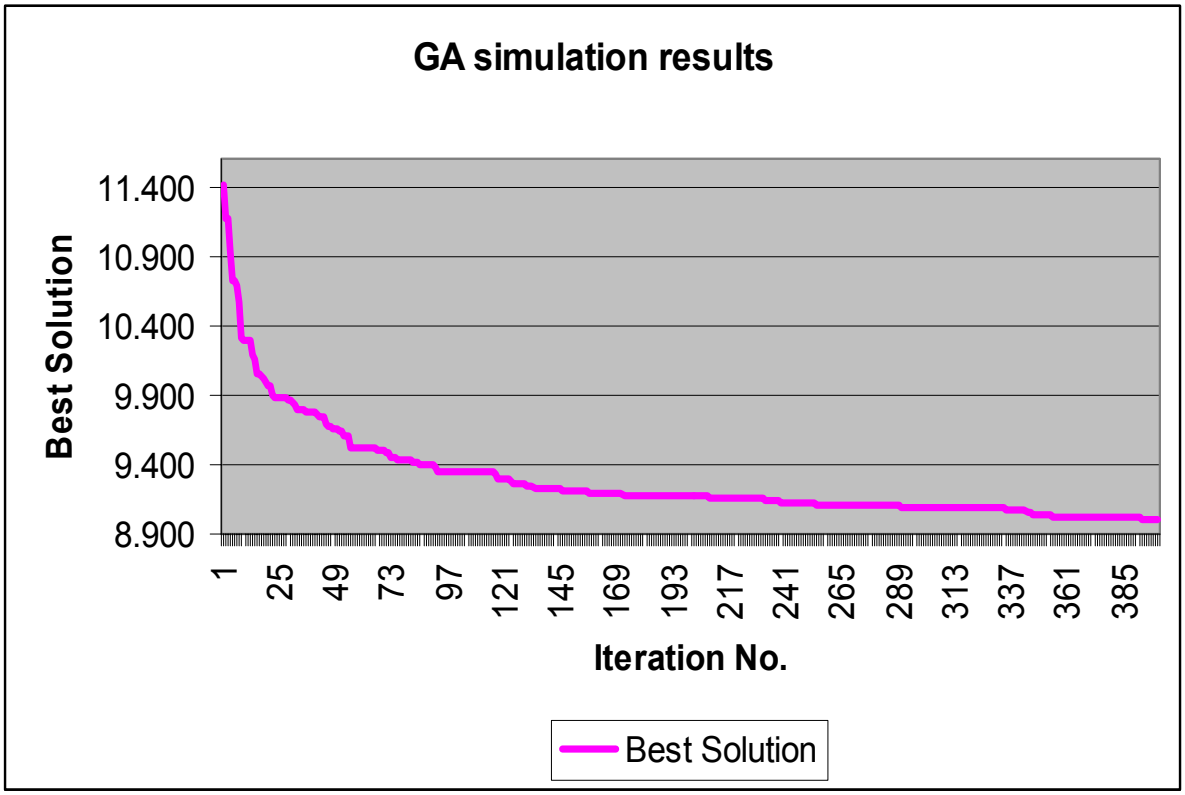


Fig A.1 GA simulation results when L/U time is 5min.