Development of a Decision Support Tool for Planning Rail Systems: An Implementation in TSAM

Chetan Joshi

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
in
Civil Engineering

Dr. Antonio A. Trani, Committee Chair
Dr. Hesham A. Rakha
Dr. Hojong Baik

December 15th, 2005
Blacksburg, Virginia

Keywords: Event Graph, Floyd's Algorithm, Schedule Delay, Nodes, Arcs, Graph, Network, Amtrak

Development of a Decision Support Tool for Planning Rail Systems: An Implementation in TSAM

Chetan Joshi

ABSTRACT

A Decision Support model for planning Intercity Railways is presented in this research. The main aim of the model is to generate inputs for the logit model existing in the Virginia Tech Transportation Systems Analysis Model (TSAM). The inputs required by the TSAM logit model are travel time, travel cost and schedule delay. Travel times and travel costs for different rail technologies are calculated using a rail network and actual or proposed rail schedules. The concept of relational databases is used in the development of the network topology. Further, an event graph approach is used for analysis of the generated network. Shortest travel times and their corresponding travel costs between origin-destination pairs are found using Floyd's algorithm. Complete itineraries including transfers (if involved) are intrinsically held in the precedence matrix generated after running the algorithm. A standard mapping technique is used to obtain the actual routes. The algorithms developed, have been implemented in MATLAB. Schedules from the North American Passenger rail system AMTRAK are used to generate the sample network for this study. The model developed allows the user to evaluate what-if scenarios for various route frequencies and rail technologies such as Accelerail, High Speed Rail and Maglev. The user also has the option of modifying route information. Comparison of travel time values for the mentioned technology types in different corridors revealed that frequency of service has a greater impact on the total travel time in shorter distance corridors, whereas technology/line-haul speed has a greater influence on the total travel time in the longer distance corridors. This tool could be useful to make preliminary assessments of future rail systems. The network topology generated by the algorithm can further be used for network flow assignment, especially time-dependent assignment if used with dynamic graph algorithms.

Acknowledgements

I am highly indebted to my advisor Dr. Antonio A. Trani for providing me an excellent opportunity and a stimulating environment for pursuing research in a topic of my choice. I would like to thank my committee members Dr. Hesham Rakha and Dr. Hojong Baik for their help and insightful ideas. I also appreciate the support of members at the Air Transportation Systems Laboratory. Finally I thank my parents; whose constant support and encouragement has made this research effort possible.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Historic Background of Rail in America

Railways as a mode of passenger transportation is one that is largely unknown or at best known only by a select few in the United States (U.S). Be it short distances or long journeys, the rail mode is hardly ever considered as an option of travel by the average traveler. Intriguing as it might sound; Railroads came to the United States of America long before the interstate highway system was even conceived. The beginning of the first railroads in the United States of America can be traced back to as early as 1824, when the government assisted in the development of the railroads in the west. The first transcontinental railroad was ordered by signing the "Pacific Railroad Act" in to law by President Abraham Lincoln in the year 1862. In the years that followed, the continuing federal bias towards railroads started to change. Passing of several acts by the government saw the railroad decline with the passage of time. In 1971, the rail passenger service act was signed leading to what is now known as "Amtrak". This service functioned on heavy government subsidies. The government faced heavy budget deficits by the end of 1994 and Amtrak was ordered to take actions towards achieving self-sufficiency by 2002. Under this pressure, Amtrak formulated what it called a "Strategic Plan", according to this plan, Amtrak focused on increasing its non-passenger revenues by expanding mail and express services. Another major development was the introduction of the first high speed rail service (the Acela express) capable of reaching speeds up to 150 miles/hour between Washington, D.C and Boston. Plans of provision of a 3000 mile high-speed rail network to be centered in Chicago, due to be implemented by 2006 were also made under the "Midwest Regional Rail Initiative" {Papacostas, Prevedouros, (2002)}. Various studies, ranging from restricted corridor studies to countrywide studies for ascertaining the feasibility of High Speed Railways in the United States of America have been performed both by researchers and professional organizations. In spite of all these efforts, Amtrak has been largely unable to capture a reasonable share in the intercity transportation market.

Table 1: Ridership on Amtrak routes (Source: GAO's Report on Financial Performance of Amtrak Routes, May1998[1]).

Passengers in thousands				
Name	**FY 1994**	**FY 1995**	**FY 1996**	**FY 1997**
Metroliners	2,025	2,001	2,011	2,081
San Joaquins	554	524	567	688
Carolinian	206	445	232	231
Piedmont	0	9[a]	29	43
Capitols	367	353	455	490
Auto Train	207	248	232	241
Northeast Direct	5,880[b]	5,871[b]	5,665	5,548
Pacific Northwest Corridor	127	268	304	335
Illini	108	101	85	89
Kansas City-St. Louis	160	143	131	156
Southwest Chief	262	255	236	257
San Diegans	1,629	1,445	1,566	1,635
Vermonter	125	96	75	85
Lake Shore Limited	328	358	352	355
Empire	1,071	1,046	979	1,057
Adirondack	85	96	95	99
Philadelphia-Harrisburg	214	198	177	215
Three Rivers	184	163	250	140
Silver Meteor	421	374	346	255
Empire Builder	453	372	310	347
Illinois Zephyr	83	82	77	82
International	116	115	110	124
California Zephyr	379	322	224	292
Capitol Limited	176	186	189	179
New York-Harrisburg	334	438	342	442
Pere Marquette	70	51	54	65
Coast Starlight	452	432	402	497
Silver Star	395	398	353	270
Silver Palm	0	0	0	188[c]
Crescent	316	270	220	247
Clockers	1,711	1,746	1,623	1,493
Desert Wind	147	120	143	80[d]
Pennsylvanian	178	233	202	160
Chicago-St. Louis	292	285	255	256
Empire-Ethan Allen Express	0	0	0	29[e]
City of New Orleans	216	195	161	174
Hiawathas	447	379	320	361
Texas Eagle	149	123	98	95
Pioneer	113	88	95	51[d]
Sunset Limited	175	161	144	124
Cardinal	107	108	80	80
Chicago-Pontiac	395	372	375	418
Gulf Coast Limited	0	0	13[f]	21[f]
Routes closed before 1997[g]	467	207	0	0
Special trains[h]	42	47	98	113
Total	**21,169**	**20,725**	**19,674**	**20,191**

[a]Service was introduced in May 1995.

[b]Includes ridership for the route between New York City and Newport News, Virginia.

[c]Service was introduced in Nov. 1996.

[d]Service was discontinued in May 1997.

[e]Service was introduced in Dec. 1996.

[f]Experimental service was introduced in June 1996 and discontinued in Mar. 1997.

[g]Includes ridership on the Atlantic City Express, Palmetto, and Hoosier routes, which were closed during fiscal year 1995.

[h]Specially contracted trains that are not part of Amtrak's regular intercity or commuter passenger service.

Table 2: Financial Performance of Individual Amtrak routes (Source: GAO's Report on Financial Performance of Amtrak Routes, May 1998[1]).

Dollars in millions

Name	Total revenues	Expenses				Profit (loss)
		Train[a]	Route[b]	System[c]	Total	
California Zephyr	35.2	64.3	10.8	3.7	78.8	(43.6)
Capitol Limited	18.8	33.6	6.6	2.4	42.6	(23.8)
New York-Harrisburg	12.3	14.8	10.7	3.0	28.4	(16.1)
Pere Marquette	2.3	3.5	1.2	1.0	5.7	(3.3)
Coast Starlight	32.2	65.5	8.6	4.1	78.2	(46.0)
Silver Star	26.3	46.9	14.4	3.7	65.0	(38.7)
Silver Palm	20.6	34.6	13.5	3.1	51.1	(30.6)
Crescent	25.8	46.7	15.4	4.0	66.1	(40.3)
Clockers	10.7	17.9	8.0	2.0	27.9	(17.1)
Desert Wind[d]	8.5	17.9	3.5	1.0	22.4	(13.9)
Pennsylvanian	5.0	7.7	4.3	1.5	13.5	(8.5)
Chicago-St. Louis	9.5	17.6	5.5	2.7	25.9	(16.4)
Empire-Ethan Allen Express	1.3	2.8	0.6	0.3	3.6	(2.3)
City of New Orleans	12.7	27.4	5.5	2.4	35.2	(22.6)
Hiawathas	9.3	15.3	6.5	5.4	27.2	(17.9)
Texas Eagle	9.6	22.7	4.6	1.5	28.8	(19.2)
Pioneer[d]	5.6	13.7	2.9	0.9	17.5	(11.8)
Sunset Limited	16.3	42.3	7.0	2.4	51.7	(35.3)
Cardinal	4.7	12.4	2.3	0.9	15.6	(10.8)
Chicago-Pontiac	10.3	23.5	10.5	3.8	37.8	(27.5)
Gulf Coast Limited[d]	0.3	1.8	0.4	0.5	2.8	(2.5)
Total	**$1,098.5**	**$1,391.5**	**$504.6**	**$151.9**	**$2,047.9**	**($949.5)**

Note: Amtrak's financial data for individual routes include only the revenues and expenses associated with providing intercity passenger service along the route. These core services are passenger-related service, mail and express service, other transportation services, and states' payments supporting certain routes.

[a]Primarily includes the train crew's salaries, fuel and power costs, all maintenance of train equipment, depreciation and debt interest for train locomotives and passenger cars, payments to freight railroads for the use of their track, and marketing and sales support.

[b]Primarily includes maintenance and depreciation for Amtrak-owned stations, track roadbed, and other facilities, as well as reservations and management support computer systems.

[c]Primarily includes staff salaries, rent, and associated expenses for corporate and SBU headquarters operations.

[d]Amtrak discontinued service on the Desert Wind, Pioneer, and Gulf Coast Limited during fiscal year 1997.

Tables Table 1 and Table 2 show the financial and operational condition Amtrak is facing.

1.2 Developments in Passenger Rail Transportation around the World

Where on one hand passenger rail is having a hard time in the United States of America, rail ridership has increased by leaps and bounds in Europe and Japan. After the introduction of dedicated high speed rail service in France and Germany, high speed rail has snatched a major part of the European intercity market share from airlines, and in cases has caused the withdrawal or discontinuation of air service in some corridors by certain companies.

The rise of the high speed railway started in the mid-1960, when Japan inaugurated its first high-speed rail service officially known as the Shinkasen and nicknamed “Bullet” that ran between Tokyo and Osaka. The bullet train had a maximum cruise speed of approximately 135 miles per hour (217 km/h). Later in the 1980’s France (TGV), Britain and Germany (ICE) started their high speed rail services. These services have evolved and have been very successful in capturing the intercity travel market. Many reasons have been given for the success of these systems in Europe and justifications for the failure of railways in USA; the most well known reason being that cities in USA have a greater sprawl as compared to those in Europe and Japan. In reality there is more behind the success of high speed rail in Europe and Japan than just land-use patterns. Taking a closer look at the functioning of these systems in Europe and Japan reveals that these systems have a very high standard of operational efficiency in Europe and Japan. This high operational efficiency is a direct result of detailed planning that goes behind the actual functioning of these rail systems.

1.3 Systems Approach to Transportation Planning

The four-step process of Transportation has the following steps:

I) Trip Generation: This process involves relating various socioeconomic and land-use variables to ascertain how many trips will be generated from a particular region or zone, depending on the scale of planning.

II) Trip Distribution: This process involves ascertaining how the generated trips will be distributed between the zones or regions in consideration. Popularly used models for

trip distribution are the Gravity model and the Fratar model {Papacostas, Prevedouros, (2002)}.

III) Modal Split: This process involves the calculation of the proportional split between different modes of transportation. Modal split is most commonly achieved by calibration of a logit model {Hensher, Button, (2000)}. The main inputs for the logit model usually are travel time, travel cost and frequency (in case of public transportation) for the modes under consideration.

IV) Trip Assignment: This process involves assignment of flows obtained from the modal split (passenger flows or vehicle flows) to the different routes that exist between origin and destination pairs.

Figure 1 shows the logical flow of the four-step process. This research is related to the calculation of travel time and travel costs for the rail mode. These travel time and travel cost values will act as inputs to the mode split step (highlighted in Figure 1) in the four step process.

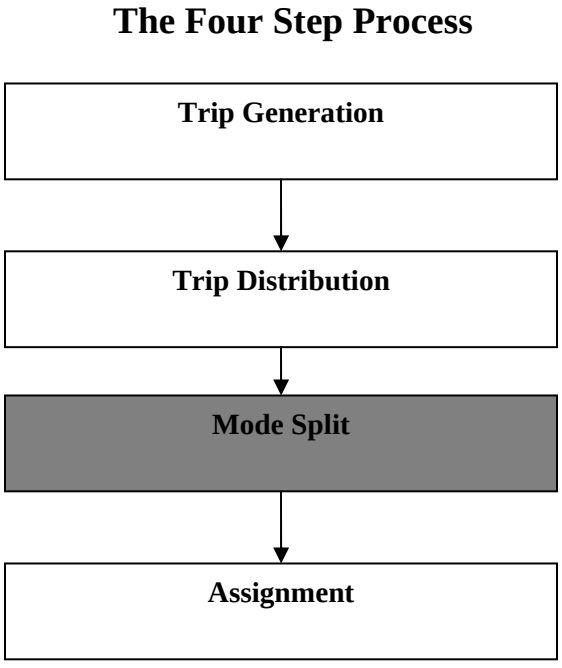


Figure 1: The Four Step Planning Process.

1.4 About the Virginia Tech Transportation Systems Analysis Model (TSAM)

The Virginia Tech TSAM is a model that uses the traditional four-step approach for air transportation systems planning. It was initially developed by the joint effort of the Air Transportation Systems Laboratory at Virginia Tech and NASA Langley, for the systems analysis of Small Aircraft Transportation System proposed by NASA Langley. The model is under continuous development and refinement. As development continues, the model is envisioned to have multimodal planning capabilities at a national level.

1.5 Scope of Work

The work done towards the completion of this thesis is a step towards making the TSAM model a multimodal intercity planning model. The scope of this research is to develop and implement a generalized and flexible methodology for building rail networks using rail schedules and estimation of travel times and travel costs for proposed high speed ground transportation technologies. The technologies modeled in this research are Accelerail 90; Accelerail 110; Accelerail 125; Accelerail 150; High speed rail and Maglev.

The methodology proposed for development of network topology is especially useful in cases where the availability of data is limited. Building a network topology is important for passenger flow assignment as well as performing network analysis. The travel times and travel costs obtained by using the model are key inputs to the mode choice module of the TSAM model.

The sample network for this study has been developed using the Amtrak schedules for year 2004. The framework of the model allows the user to change frequencies, routes/lines, average dwell times and stations, and perform scenario analysis to evaluate existing and proposed or future routes.

2 Literature Review

Analysis of high speed rail feasibility has been carried out by researchers throughout the United States of America. These studies have mostly been carried out in order to investigate the feasibility of high speed rail in specific corridors, and were part of a cost comparison with other competing modes of intercity transportation mainly air and automobile.

“Eurailspeed 98” is a study for the analysis of the economic impacts of Florida high-speed rail carried out by Lynch and Picq (1997) is a detailed analysis performed for the Orlando-Tampa-Miami corridor. The modeling for demand and rider-ship for the study was carried out by SYSTRA consulting; a Europe based multinational specializing in railway planning and operations modeling. The study was primarily driven by the looming problem of highway congestion in Florida, which is attributed to the substantial increase in the population of the state. Taken into account in the study were the possibilities of induced demand and that of code-share (Air-Rail co-operation) operation. Mode split and market shares were also projected in the study. It was found in the study that introduction of a high speed rail service would result in; a reduction of 1.4 million auto trips and 60 thousand fewer aircraft flights; an increase in employment; increased economic activity of $1.667 billion per year in the high speed rail deployment period; reduction in travel times and decrease in emissions.

Levinson, Kanafani and Gillen {Levinson et al, 1998}, carried out a cost comparison study between air, rail and auto in the California corridor. The analytical framework of their study breaks down the full cost of transportation into infrastructure costs that include capital costs of construction and debt service, and costs of maintenance and operating costs as well as service costs to government or private sector; carrier costs that is the aggregate of all payments by carriers in capital costs to acquire a vehicle fleet, maintain and operate that vehicle fleet, less costs such as usage charges which are transfers to infrastructure, these costs were labeled as carrier transfers; aggregate of all fees, fares and tariffs paid by users in order to purchase a vehicle, and cost of maintaining and operating the vehicle were clubbed under user money costs; user travel time costs were defined as the amount of time spent traveling under both congested and

free flow conditions multiplied by the monetary value of time; social costs were assumed as the additional net external costs to society due to emissions, accident and noise. Based on their analysis, the authors found that air had the cheapest full cost whereas auto and rail had almost the same full cost. Auto was found to have the highest external cost. Also mentioned was the fact that if rail ridership were to be greater than predicted by their model, the per capita cost of rail would reduce significantly.

Spiess and Florian {Spiess, Florian, 1993} introduced the concept of an exploded network. As per this approach, the set of nodes not only contains the physical nodes of the underlying street or rail network, but it also contains an additional node for each transit stop of each line. The links are accordingly subdivided into various classes, such as boarding, alighting, in-vehicle and walking links. In their approach, frequencies are given to all links, and since only boarding links have waiting involved, they have a finite frequency that takes the value of some composite cost. All other links have an infinite frequency so that the penalty is minimized. A similar idea is developed in this research, the explanation for which has been described later.

In another report written by the United States General Accounting Office (GAO) addressed to Senator Ron Wyden; it has been pointed out that National network analysis for the rail network was carried out by Amtrak as part of their effort to achieve self-sufficiency. According to the report, Amtrak's uses mode split models only for short, high speed and high frequency routes that are less than 500 miles, Amtrak calls such routes "corridors". Most feasibility studies have been concentrated on corridor analysis. The GAO's report attributes Amtrak's failure to attract demand and improve service under its "Network Growth Strategy" mainly to the following two reasons:

(1) Amtrak overestimated revenues expected from new mail and express service.

(2) Amtrak could not reach an agreement with freight railroads over capital funding and other implementation issues.

A study published by US Department of Transportation and the Federal Railroad Administration was focused on evaluation of different (high speed and very high speed) rail options for USA. The study was conducted with the support government agencies and private consultants. Six technology types were analyzed in the study; Accelerail-90;

Accelerail-110; Accelerail-125; Accelerail-150; High Speed Rail (186 mph option) and Maglev (300 mph). The focus of the study is on analysis of system requirements and performance of various high speed ground transportation options and evaluation of the respective benefits and costs of the options.

2.1 Network Analysis and Graph Theory

The use of graph theoretic concepts has become widespread in the field of transportation networks these days, be it roads, railways, airlines or pipe networks, graph theory has found application in it. This can be attributed to the fact that we have more computational power at our disposal these days than before and this trend is still growing. Also, the interdisciplinary nature of transportation systems engineering has been beneficial in that there have been valuable contributions to the field by researchers from other fields like mathematics, statistics, computer science and operations research. Some basic concepts and definitions of graph theory with context to transportation networks are discussed next.

A network when symbolically represented can be said to be a graph. This graph is composed of nodes represented commonly by V, and edges commonly represented by E. Hence a graph G with V nodes or vertices and E edges is referred to as G (V, E). In the physical sense of a transportation network, a node v may represent an intersection, a station or an airport in case of a road, rail or airline network respectively. An edge e may represent a road, rails or flight path and normally has a length associated with it. In Figure 2 ; the graph G (v, e) has vertices v = (1, 2, 3, 4) and edges e = {(1, 2), (1, 3), (4, 2), (4, 3)}.

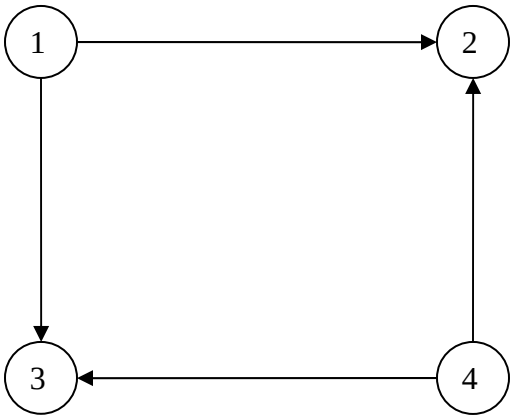


Figure 2: Graph G(v, e).

The graph in Figure 2 is a directed graph, meaning that the direction in which flow might take place is restricted. A sub-graph is a subset of a graph, for example in the above graph G, G* is the sub-graph of G containing nodes (1, 2). Naturally the union of all the sub-graphs of G will result in the graph G. A sequence of movement via intermediate nodes leading from an origin to a destination may be called a path. More often than not, finding the shortest path in the network is required. Solving this problem is critical from the stand point of assigning flows in the network during the assignment stage of the four step process of transportation planning. There are various algorithms available for solving the shortest path problem. Two of the most commonly used shortest path algorithms are discussed next.

2.1.1 Dijkstra's Algorithm

Dijkstra's algorithm is generally used in cases where one to all shortest paths is required to be found in a weighted graph. Dijkstra's algorithm {Weiss, (1992)} is an example of a greedy algorithm, since it works by solving the problem in stages. In order to find the shortest path, a vertex or node v that has the minimum distance from the specified source or origin is selected and is tentatively declared as the shortest path. Any node that is not directly connected to the source node is initially assumed to have a distance of infinity. As the process passes through stages if a shorter distance from one node to the other is

found it is updated and the predecessor node is also updated in the shortest path queue. The pseudo code for Dijkstra's algorithm is given in Figure 4.

```
void
init_table (vertex start, graph G, TABLE T)
{
 int i;
 read graph (G, T);
 for (i = NUM_VERTEX; i > 0; i--)
        {
         T[i].known = FALSE;
         T[i].dist = INT_MAX;
             T[i].path = NOT_A_VERTEX;
        }
        T[start].dist = 0;
}
```

Figure 3: Table Initialization.

```
void
dijkstra(TABLE T)
{
        vertex v, w;
        for (; ;)
        {
                V = smallest unknown distance vertex;
                if (v == NOT_A_VERTEX)
                        Break;
                T[v].known = TRUE;
                for each w adjacent to v
                        if ( !T[w].known)
                        if( T[v].dist + c_{v,w} < T[w].dist )
                        { /* update w */
        Decrease( T[w].dist to t[v].dist + c_{v,w} );
                                T[w].path = v;
                        }
        }
}
```

Figure 4: Pseudo-code for Dijkstra's Algorithm.

2.1.2 Floyd's Algorithm

Floyd's algorithm {Weiss, (1992)} is used when we are interested in finding all-to-all shortest paths. This algorithm is an implementation of dynamic programming just like Dijkstra's algorithm. It works by successively inserting a node between two nodes and checking if the distance obtained by this insertion is shorter than the existing distance, if the distance after insertion is found to be shorter, the node is inserted as a predecessor in the precedence matrix and the distance is updated, otherwise the existing distance is retained and no insertion is performed in the precedence matrix. The pseudo code for Floyd's algorithm is shown in Figure 5.

```
void
all_pairs( two_d_array A, two_d_array d, two_d_array path )
{       int i, j, k;
        for (i = 0; i < |V|; i++)
                for (j = 0; j < |V|; j++)
                        {
                        D[i][j] = A[i][j];      /*  D is the distance matrix, A is the
                                                /*  adjacency matrix
                         Path[i][j] = NOT_A_VERTEX;
                        }
                for ( k = 0; k< |v|; k++)
                        for ( i = 0; i < |v|; i++)
                                for ( j = 0; j < |v|; j++)
                                 if ( d[i][k] + d[k][j] < d[i][j] )
                                        /*update min */
                                        {
                                          d[i][j] = d[i][k] + d[k][j] ;
                                           path[i][j] = k ;
                                        }
                                }
                }
        }
}
```

Figure 5: Pseudo Code for Floyd's Algorithm.

2.1.3 Label Constrained Problems

When complex real world problems are considered, especially those encountered in the case of transportation networks, the discussed generalized algorithms can not be applied directly. In many cases there are constraints that fall beyond the framework of these

algorithms. In such cases, modification or tweaking of these algorithms might be required. A classic example of such a constraint based shortest algorithm is the Label constrained shortest path algorithm. Label constrained algorithms have found applicability in cases where multimodal shortest paths are required to be found. Marathe and Barret {Barret et al., (1998)} in their approach have used an approach where a mode choice constraint is imposed by using a label string with a character for each mode in a labeled transportation network. Edge labels describe the allowable moves a person may make. A nondeterministic finite automation (NFA) of the network and mode choice is performed after which a cross product of these NFA's gives a set of feasible path traversals. An NFA defined as a five tuple $(S, \Sigma, \delta, s_o, F)$ where S is a finite nonempty set of states, Σ is the input alphabet, δ is a state transition function and s_o is the initial state and belongs to the set S. The main applications of the label constrained method include trip chaining, finding k-similar paths and giving turn penalties to traffic that amounts to decision sensitive costs.

2.2 Rail Planning Software

Some of the currently available commercial software packages for rail planning are discussed in this section. Common applications of these software packages include advanced line capacity planning and rail operations scheduling (timetabling).

2.2.1 RAILSIM

RAILSIM, a railway planning model developed by SYSTRA consulting is an example of a model that allows performance based schedule development and planning of new rail facilities. The base software consists of a Train Performance calculator that has inbuilt rolling stock libraries, resistance equations and a report generator. Additional modules in the software package include a track profile generator, load flow analyzer (for AC and DC) that allows modeling of electrical networks, a Network simulator, headway calculator and breaking distance calculator, signal design add-on and an interactive editor that has AutoCAD compatibility. The user can interactively code the network as per requirement. RAILSIM is a good model for detailed operational planning of railways and has been applied in projects like the Caltrain "Baby Bullet" Express Service, Metro-North

Railroad New York State Territory Database, NYCT Manhattan Bridge and PATH World Trade Center Service Restoration. More details on the software are listed on the website http://www.railsim.com .

2.2.2 MultiRail

MultiRail has been developed by Multimodal applied systems, the model has capabilities that include timetabling, publishing, line capacity/conflict management, time-distance analysis, crew planning, equipment cycling, and platform management capabilities. The additional tools available with the software give it capabilities to examine issues concerning passenger yield management. As far as network capabilities are concerned, the software allows interactive development the rail network, that takes into account the geographical and performance attributes. A rail network can also be imported from an outside source. After importing the network, the Track Manager can be used for defining detailed track and switch characteristics that include the length, maximum allowable speed, switch type, headway requirements, and grade and curvature of the track.

2.2.3 VISUM

This product is part of the PTV-VISION (Germany) transportation software suite. VISUM can be used for analyses and planning transportation networks with a special functionality for public transportation (Rail transit networks). VISUM addresses transportation planning issues such as Multi-modal transportation modeling, continuous control that aids in monitoring the performance of a transit system, modeling of fare systems, modeling of large networks at the national level and also the European level. VISUM has an inbuilt demand model called VISEM; this model generates trip chains based on activity chains.

The network model in VISUM has two categories, private transport and public transport represented by PrT and PuT respectively. The speeds on the network in the model are restricted by network capacities and transit speeds are derived by timetables (pre-formulated). In the integrated network, nodes may represent intersections or public transit stops. Specific links in the network can be used only by specific type of vehicles on the network. The model allows for usage of links as per the type of vehicles, for example the rail vehicles may use only the links defined as rail network etc. Creation of a network can

be done using digital link network data, for creation of multimodal networks, relational database approach is used that matches and geocodes transit stops that results in the network connectivity. The software allows for construction and editing of transit lines interactively, here a user or planner can test what if scenarios. After the transit line has been created, a routine optimizes the timetable to minimize transfer time.
The VISUM approach is shown in Figure 6.

2.3 Contribution

The proposed model will act as a complete planning tool when combined with the Transportation System Analysis Model (TSAM). After complete integration with TSAM, it will enable the user to plan for three modes (Air, Rail and Auto) in one model. No model currently exists for such planning at national level. The model will allow for network analysis with complete modeling of routes including transfers. The effect of corridor improvement on the entire network can also be evaluated using this model. The model allows analysis of both existing rail systems by using schedules and future rail systems by route, frequency and dwell time specification; this is of particular help in studying technology up gradation on existing or even new user defined routes. The transfers and connections are evaluated using an exact method rather than a heuristic method.
In formulation of the connectivity of the network, a multigraph is converted into a simple graph; as a result the problem is made to fit into the framework of existing shortest path algorithms. The methodology developed in this research can also be extended for solution of problems such as intermodal shortest path and passenger flow assignment.

Assignment based on lines	Assignment based on timetables
1. route search	**1. connection search**
Search for best route: impedance = access time + egress time + in-vehicle time + transfer penalty P x no. of transfers + mean transfer time (= Fac x mean headway) Repeat search with different penalties P and weightings of Fac to determine several routes	Search for best connection impedance = access time + egress time + in-vehicle time + transfer penalty P x no. of transfers + actual transfer time Repeat search for all possible departure times at origin stop
2. route choice	**2. connection choice**
Delete unattractive routes, where journey time > min. journey time x factor transfers > min. transfers + factor	Delete unattractive connections, where journey time > min. journey time x factor transfers > min. transfers + factor
3. route split	**3. connection split**
For each route calculate • perceived journey time PJT which consists of weighted components of journey time • fare • Impedance Imp = f (PJT, Fare)	For each connection calculate • perceived journey time PJT which consists of weighted components of journey time • fare • temporal utility U which results from comparing the desired departure time of passengers with the actual departure times of the connection • Impedance Imp = f (PJT, Fare, U)
Distribute trips with Kirchhoff Law $P_i = \frac{Imp_i^{-\alpha}}{\sum_{j=1}^{n} Imp_j^{-\alpha}}$ P_i proportion of trips using route/connection i n number of routes/connections Imp_i impedance of route/connection i α impedance sensitivity factor	

Figure 6: VISUM Assignment Approach.

Source: Friedrich, M., Haupt, T., Nökel, K. (1999) Planning and Analyzing Transit Networks –An Integrated Approach Regarding Requirements of Passengers and Operators, *Journal of Public Transportation*, Volume 2, No. 4, p.19-39.

3 Problem Description

The primary objective of this research is to develop and integrate a user friendly systems planning tool for the rail mode in the Virginia Tech TSAM model. The user is allowed to modify key elements such as technology type (Accelerail 90, 110, 125, 150; high speed rail and maglev), frequency of departures and network connectivity with the capability of induction of new station sets in order to assess the feasibility of a particular service.

The development of the model is further decomposed into three parts;

1. National network analysis of the existing U.S passenger rail network using Amtrak schedules.
2. Development of a performance based travel time model for Accelerail 90, 110, 125, 150, high speed rail and maglev.
3. Development of a technology based travel cost model for Accelerail 90, 110, 125, 150, high speed rail and maglev.

4 Methodology

The shortest travel times and the corresponding travel costs for all station pairs were obtained by the sequential execution of four steps;

1. Network connectivity and distance was obtained from Amtrak schedules
2. Travel time was loaded on the distance matrix by using the train performance function and travel time assignments were made to dummy links
3. Floyd's algorithm was used to find the shortest path between all pairs of stations using total travel time[1] as criteria
4. The travel costs for all pairs of stations were evaluated using the distance based cost model

The conceptual flow of the processes involved in achieving the above stated goals is shown in Figure 7.

[1] Total travel time is the combination of schedule delay, line haul time, layover/transfer time and dwell time

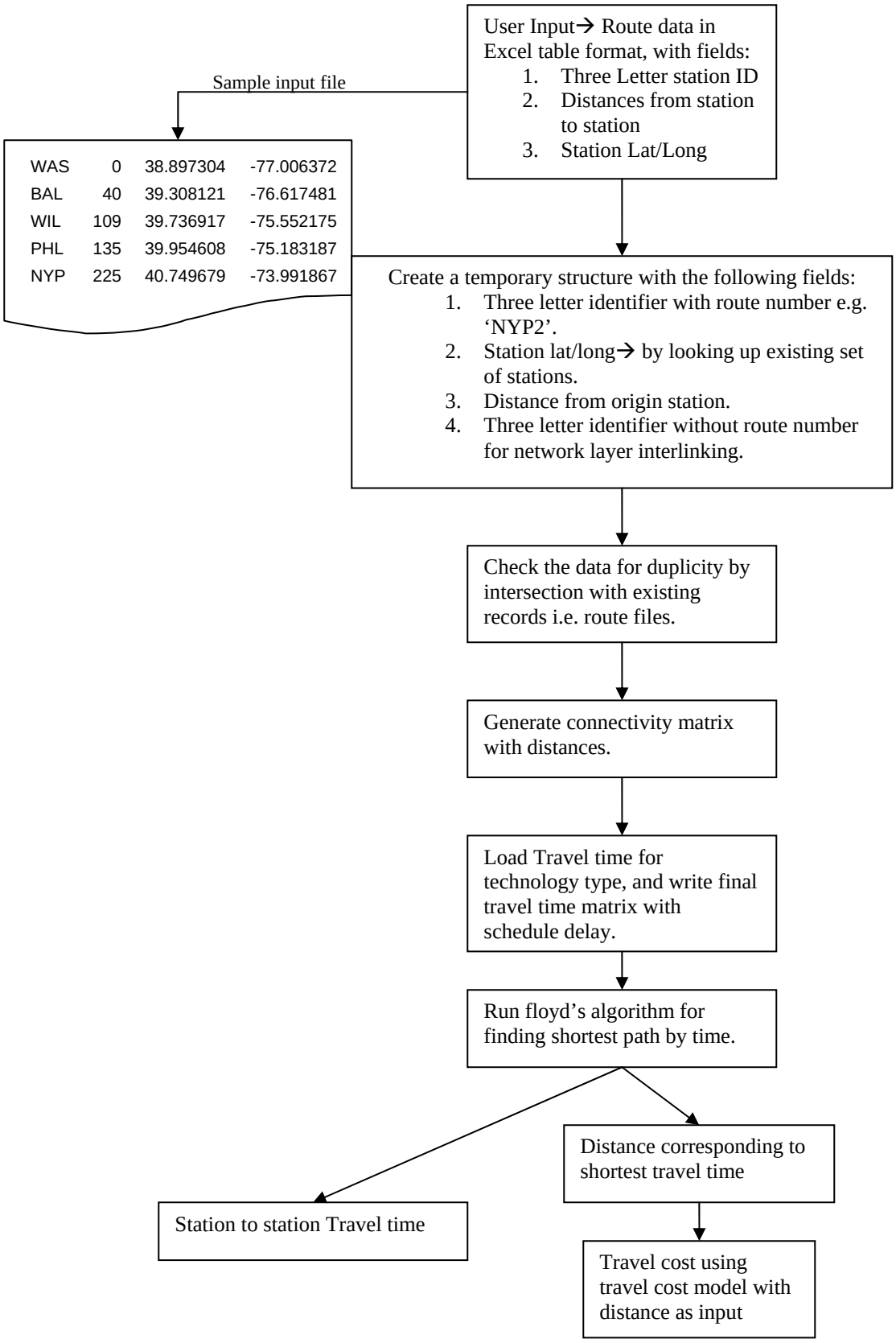


Figure 7: Process Flow for Complete Analysis.

4.1 Obtaining Network Connectivity and Distance for the Station Sets

There are a total of 923 railway stations in USA and Canada as per the NTAD. Only the stations indicated as ACTIVE by the NTAD were first selected for analysis. Data matching with the Amtrak station data and filtering led to a final station set of 464 stations across the United States of America. Station to station distances data from Amtrak schedules(available as PDF files) were then pasted in Excel and imported into Matlab and stored as an array of structures that consisted of six fields viz. Station Code, Station Latitude, Station Longitude, Distance, Frequency and Dwell time. A major problem in the creation of the structure was that Amtrak schedules do not have the station codes listed in them. This necessitated the replacement of station names by three letter station codes which were listed in the NTAD. Merging and replacement was a cumbersome task since the NTAD in many cases does not have exactly the same station names as the Amtrak schedules; as a result, using the command 'strcmp' (string compare) did not return a logical value of 1. A lot of rule based and manual filtering was thus needed to arrive at the actual three letter station codes in the route structure. Once the routes were ready, the connectivity matrix was generated from these routes; the methodology for connectivity establishment has been explained next.

Since dedicated passenger lines in two directions were assumed, the railway network represented an undirected graph *G = (V, E) (which gets converted into a mixed graph after dummy link insertion)*; *V* being the vertices or nodes and *E* being the railway lines or edges. The vertices or nodes in this particular case represented stations, since a low resolution network is modeled. A high resolution model would also include critical points and crossings and switches. High resolution networks are required where one is interested in investigating the safety of railroad crossings. This aspect is beyond the scope of this study.

A route or schedule was assumed to represent an array of nodes $\{v_1, v_2 \ldots, v_n\}$ where $v_i \in V$, $i = 1 \ldots n$ that constituted a particular route or schedule. A connected link or edge on the network was thus (v_i, v_{i+1}). Treating each route separately, a number of independent graphs were obtained. In order to connect these independent graphs, dummy links were inserted into the network. Insertion of dummy nodes and overall connectivity

definition was achieved by using a relational database approach where table elements were accessed by indexing.

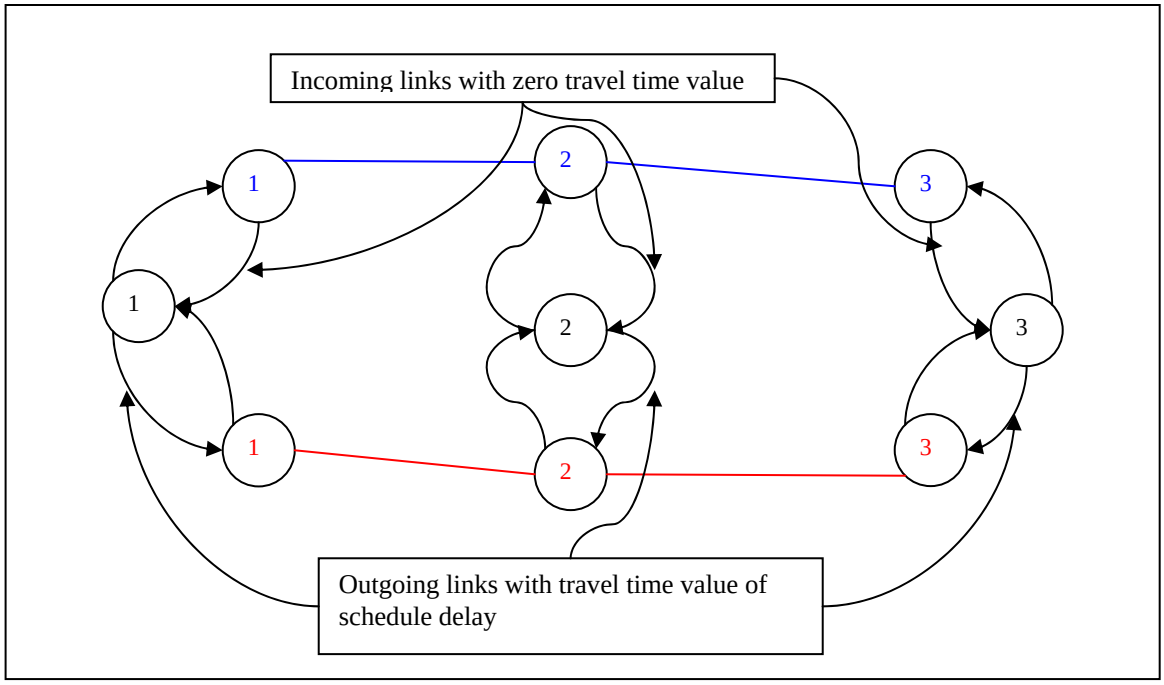


Figure 8: Representation of Rail lines as Nodes and Links/Edges.

In Figure 8; if nodes 1, 2 and 3 were to be served by two different trains with one train having a frequency x and the other having frequency y. Assuming the trains to have the same performance characteristics, the line haul time of both the trains would be the same. When frequencies are considered though, the trains might have different total travel times since schedule delay (discussed later) is a function of frequency. Using the red train/route would lead to a schedule delay of 4 hours if a frequency of 1 was assumed and using the blue route would lead to a schedule delay of 1 hour assuming if a frequency of 4 was assumed. If link travel times were the same, the blue route will on an average get the passenger from 1 to 3 faster. The links connecting the same number nodes, 1, 2 and 3 do not really exist physically but are dummy links that take the value of schedule delay. Thus, these links perform the function of connecting the red graph with the blue graph and act as penalty links. At node 2, assuming that the transfers to be feasible; a traveler may choose to either stay in the same train/route or get into a different train to get to his/her destination. In the case presented above, it is obvious that the passenger will opt

not to transfer since this will lead to much greater travel time for him/her. If there were an additional route that went from node 2 to 3 with a high frequency and a much lower travel time, the total travel time with a transfer at node 2 might become less than the route that does not involve a transfer. This phenomenon can be easily captured by such an explosion of the network.

From a graph theoretic point of view, a multigraph is transformed into a simple graph; Figure 9.

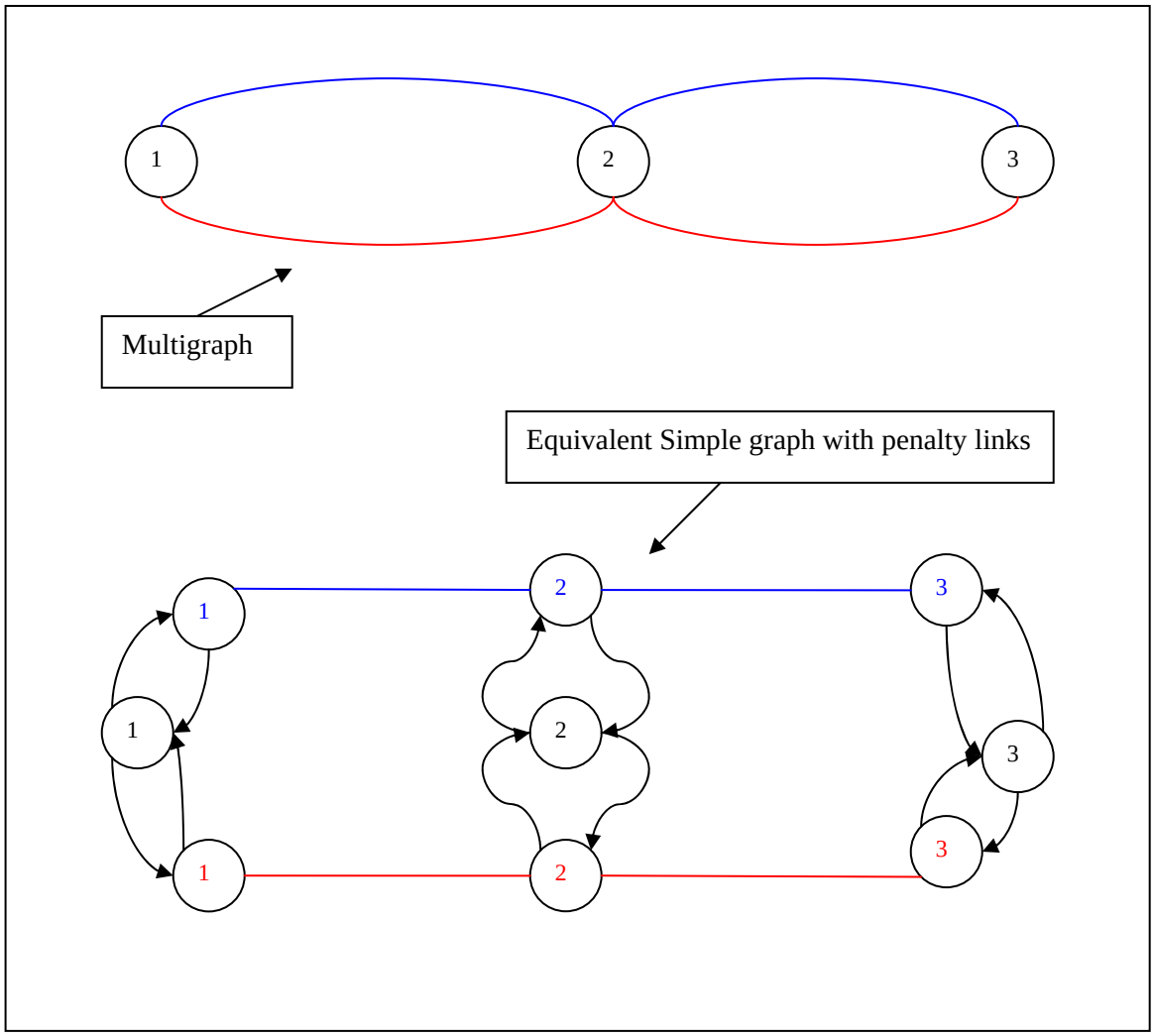


Figure 9: Conversion of Multigraph into Equivalent Simple Mixed Graph.

This fact was used to formulate an algorithm that gave the connectivity in a network using schedules. The assumption was that trains would operate back and fourth on routes but in general this may not always hold true as sometimes trains run around in cycles. The steps involved in the execution of this algorithm have been shown in Figure 10;

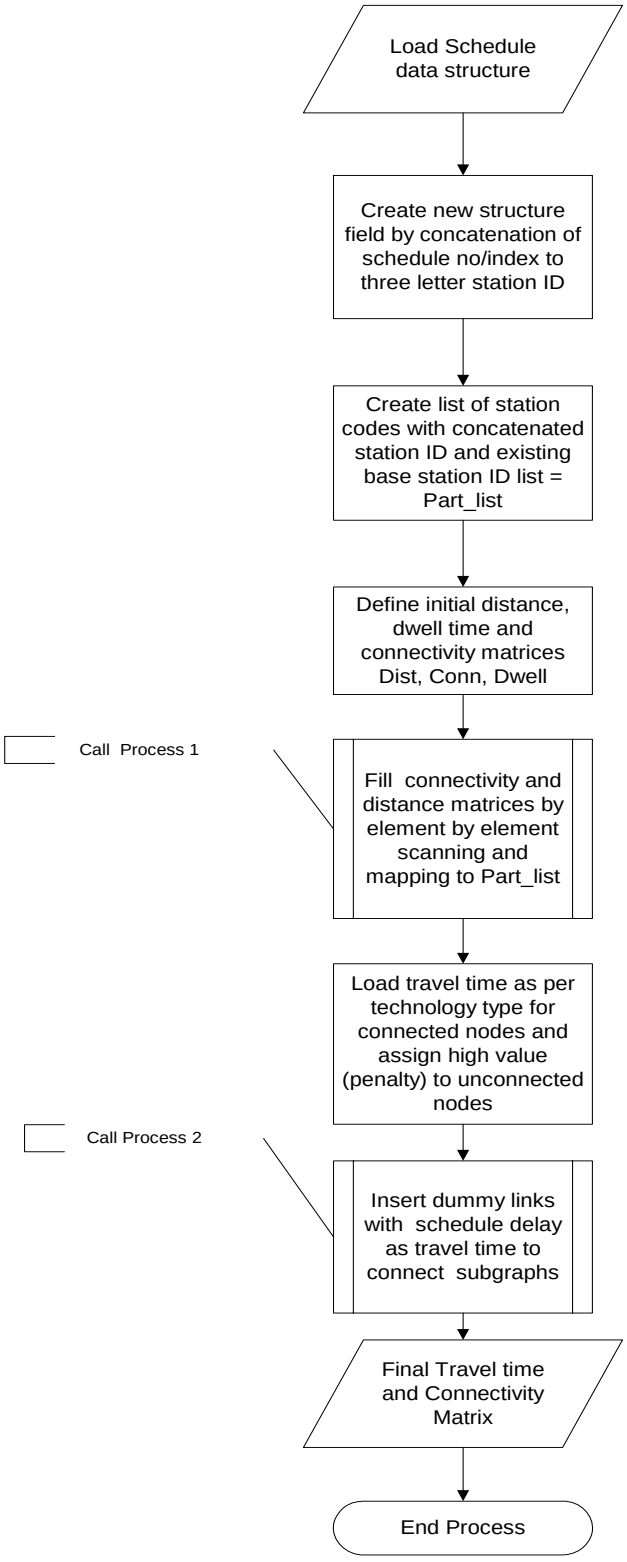


Figure 10: Flowchart for Connectivity Extraction using Schedules, Processes 1 and 2 are Explained in Figures 11 and 12 Respectively.

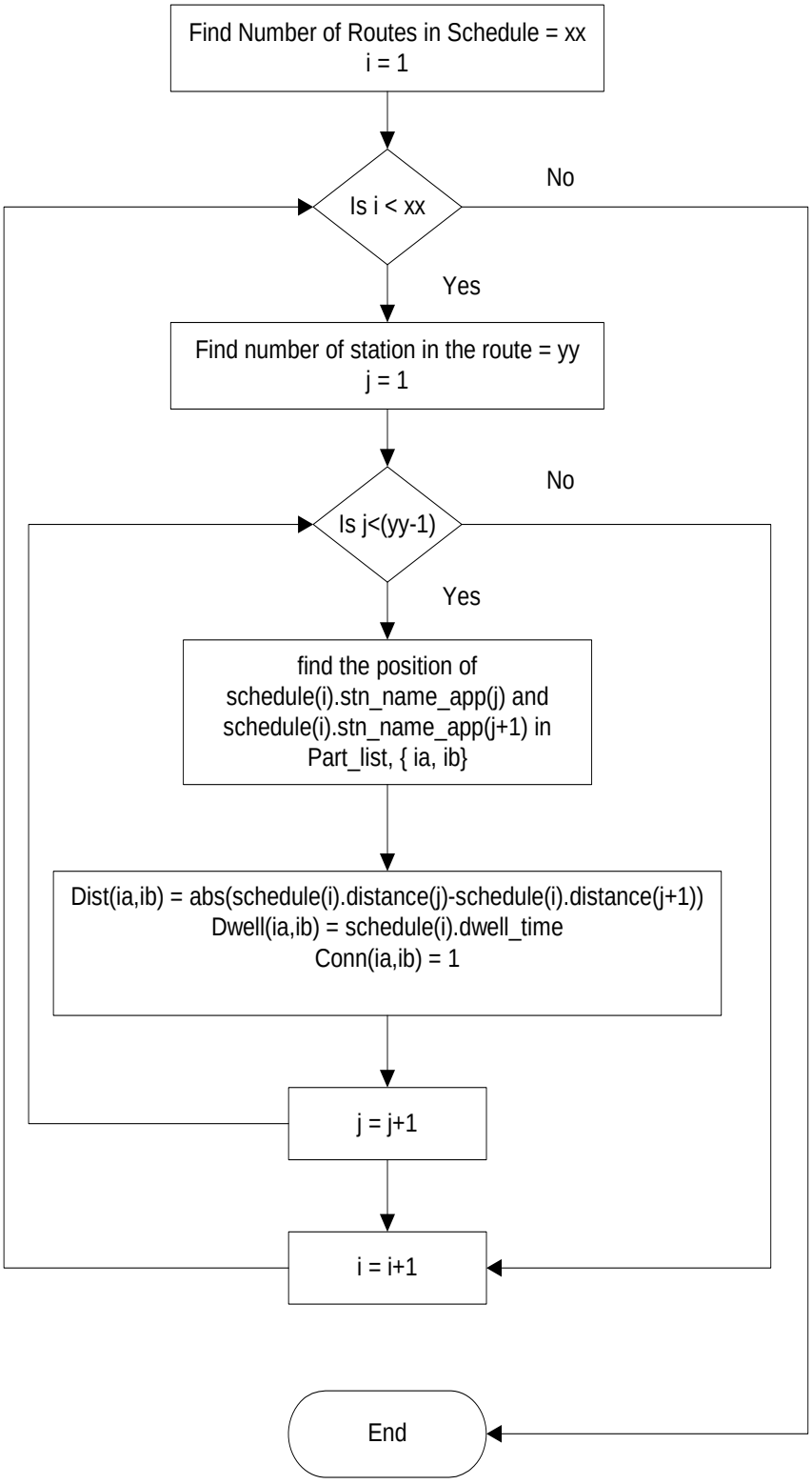


Figure 11: Flowchart for Process 1 Shown in Figure 10.

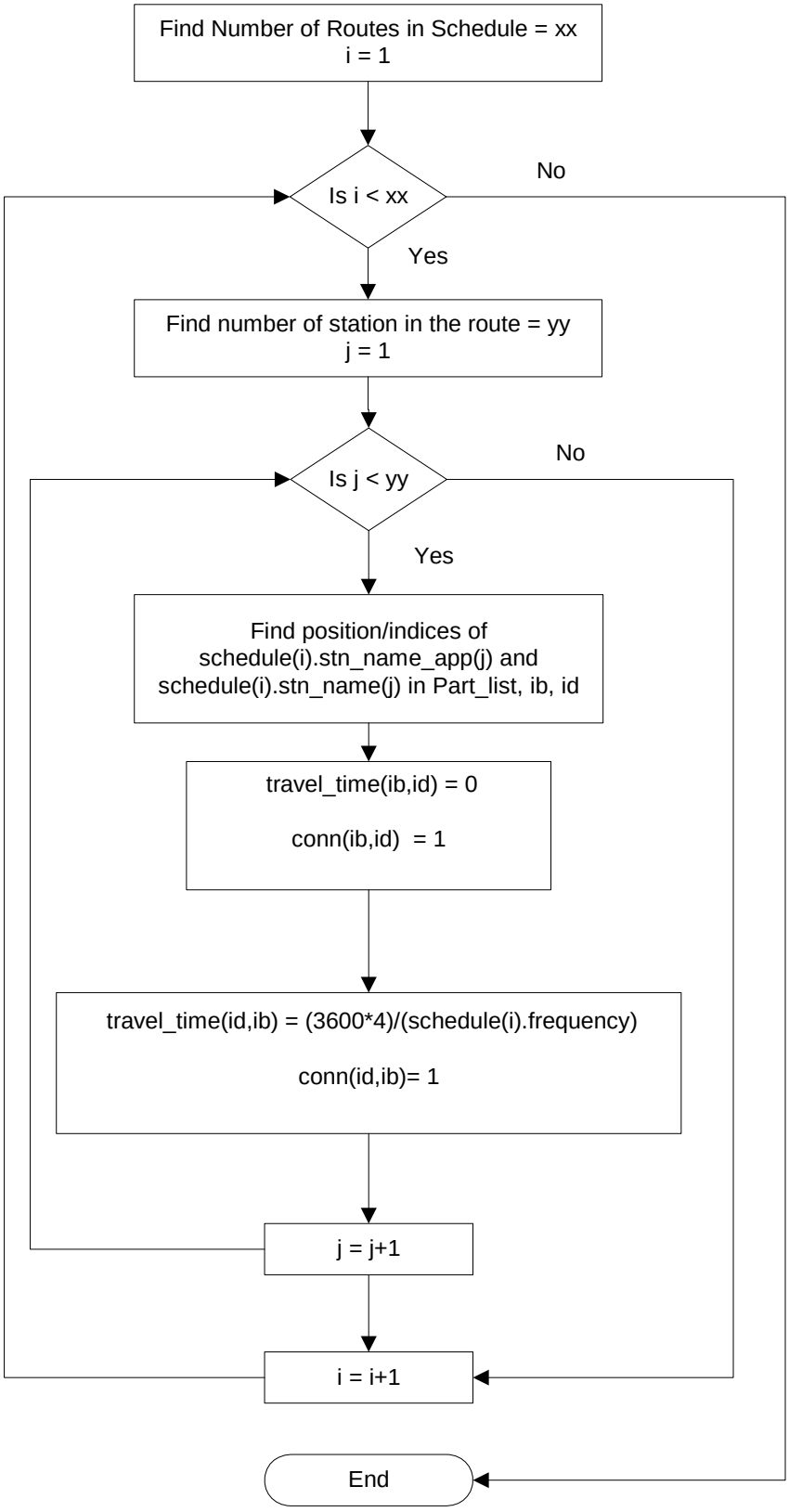


Figure 12: Flowchart for Process 2 Shown in Figure 10.

The distance matrix for the connected nodes was extracted on the same principle. A distance penalty value of 9999 was assigned where the nodes were not connected directly.

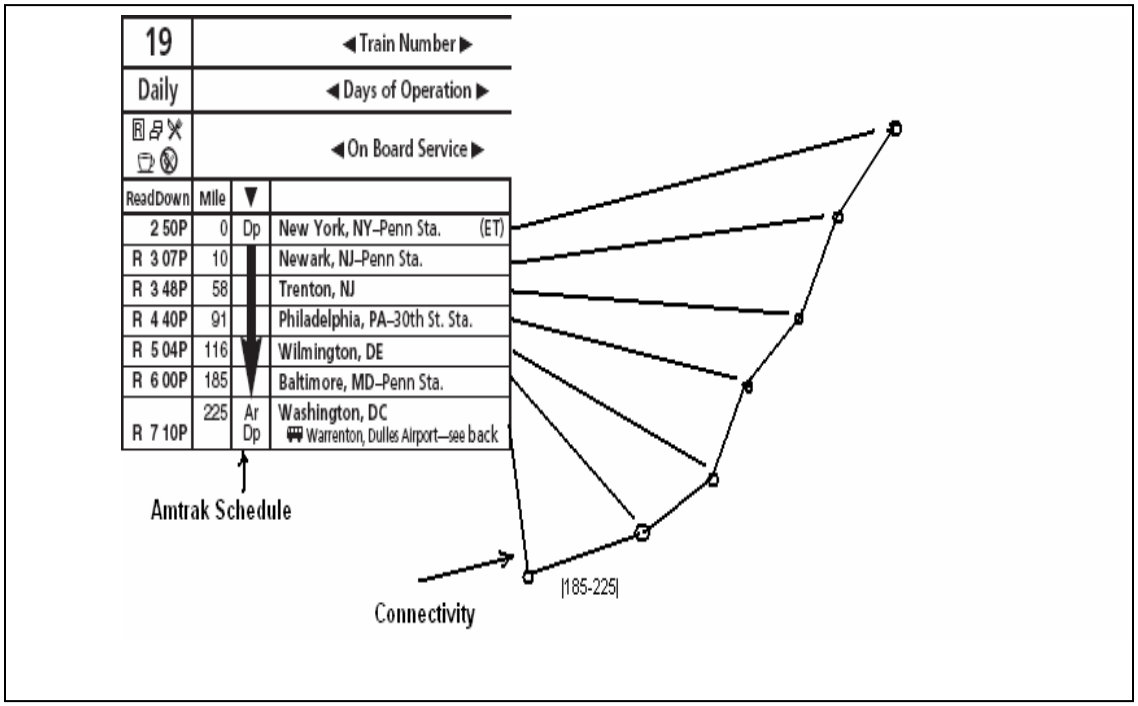


Figure 13: How Schedule Data Translates into Connectivity.

4.2 Finding Shortest Paths between all Pairs of Stations

Floyd's algorithm {Weiss, (1992)} was used in order to find the shortest path between all pairs of stations. Floyd's algorithm was used since it is easy to code and implement for an all to all case, it has a run time of $O(n^3)$, while running the usual Dijkstra's algorithm would require a run time of $O(n^4)$. Here n is the number of vertices or nodes. Another way of implementation of Dijkstra's algorithm is by using priority queues, this makes Dijkstra's algorithm as fast as or faster than Floyd's algorithm when practical run time is considered for sparse graphs. The coding in this case is more involved. In Floyd's algorithm, the shortest paths are stored in the form of a precedence matrix. Since many points did not have direct service, it was necessary to find the points at which passengers would need to change trains. Even though the precedence matrix gave paths between all

O-D pairs, it did not directly give information about 'Transfer Points' where the passenger would need to change trains. Finding transfer points was a key element of the analysis. Finding the transfer stations was achieved using the following set theory relationships,

Let D = List of dummy node stations

P = Nodes on the shortest path

Now if, $P \cap D = X$

Here X is the vector of dummy nodes on the shortest path, further in this vector if Y $=\{x_2 .....x_{n-1}\} where, x_i \subseteq X$. Y is the vector that represents the transfer stations.

4.3 Travel time calculation

As in any public transportation system, the total travel time in case of rail is composed of a) Line haul time b) Access and Egress time c) Dwell time d) Transfer time/Layover time if applicable and e) Schedule delay.

4.3.1 Line Haul Time Estimation Approach

Two approaches were considered in order to evaluate acceleration characteristics of the train-sets. The first approach was to use a vehicle dynamics based model and the second approach involved using a linear acceleration decay model.

4.3.1.1 Vehicle Dynamics Model

The basic equation of motion can be derived based on the second law of Newton; according this law, mass, acceleration and force are related in the following manner;

F = mass * acceleration (m a)..... 1

Where,

F = Total force on the body (Newton)

m = mass of the body (Kg)

a = acceleration (m/s^2)

The total force F can be broken down into tractive force or effort (TE) and resistance force (R); the resistance force again consists of various components or forces (discussed later). Thus, we get:

(TE-R) = m a..... 2

Also, since acceleration (a) is given by,

a = dv/dt 3

TE – R = m. dv/dt 4

Integrating the above equation, gives velocity v of the train at time t. This can be used to obtain a v-t plot.

Resistance to motion R comprises of a set of many forces, these forces have been summarized below:

1. Vehicle resistance

i) Rolling resistance: Resistance at the contact of the wheel and surface is rolling resistance. Its value depends on the smoothness and hardness of the wheel and support surfaces and the weight on the wheel.

ii) Way or track resistance (this is a function of track position, nature of surface, joints in the rails, sway and oscillation and concussion): It depends mainly on the condition, characteristics and composition of the track.

The values for these resistances can't be measured with precision and thus have been established by experimental studies (by Davis for railroad vehicles). These basic values of resistance depend on weight of the vehicle. The expression for resistance due to rolling and way resistance developed empirically by Davis is given below:

$R_r = (c_1 + c_3/p + c_2V)*G$..... 5

Where,

R_r = Resistance in Newton's

c_1, c_2, c_3 are coefficients

p = axle loading (weight per axle)

G = Weight of the vehicle

V = speed of the vehicle

iii) Aerodynamic resistance

Aerodynamic resistance is the function of the following:

1. Projected area
2. Shape and surface characteristics

3. Density of the medium(air in this case)
4. Velocity of the body

The basic expression for air resistance is given by: $c_d.\rho.A.V^2/2$

Where,

c_d = Coefficient of drag, ρ= density of the medium

V = Velocity of the vehicle

A= Frontal or projected area

By substituting the known values of constants we obtain,

$Ra = c_a.AV^2$ 6

An expression for total vehicle resistance for rail vehicles has been formulated by the Association of American Railroads,

$R = (0.65 + 129/p + 0.009V)*G + 0.0716V^2$ 7

2. Alignment resistance

i) Gradient

From the diagram below it can be clearly seen that the grade resistance is given by the horizontal component of the force G; thus, we get

$R_g = G$ *sin a*..... 8

But, at small values of a, *sin a* approaches *tan a*. The grade resistance may thus be written as,

$R_g = G \tan a$..... 9

ii) Curvature

The expression for curvature resistance has been empirically derived. Hamburger Hochbahn AG, Hamburg has developed the diagram of curvature resistance

Co-efficient r_c as a function of radius of curvature K (m) for their latest coaches. The curve resistance is given by:

$r_r = 650/(K-55)$..... 10

$Rc = rc*G$ 11

Where,

K is the radius of curvature in meters.

Expression for Total resistance:

Rt = Sum of all components of resistance

Or

$Rt = (c_1 + c_3/p + 10*i + c_2*V)*G + c_aAV^2$ 12

Where,

i = gradient in percentage and other terms have their usual meaning (explained earlier).

Velocity profile of the train under varying conditions

The following velocity profiles of the train-set were obtained by using MATLAB. The Ode function was used and an event location option was used to obtain profiles for multiple stops.

Data Used for the train-set:

Max Adhesive weight = 204 tonnes

Total weight = 752.4 tonnes

Power = 5000 hp

c_1= 0.65

c_2= 0.009

c_3= 129

Load per Axle = 17 tonnes

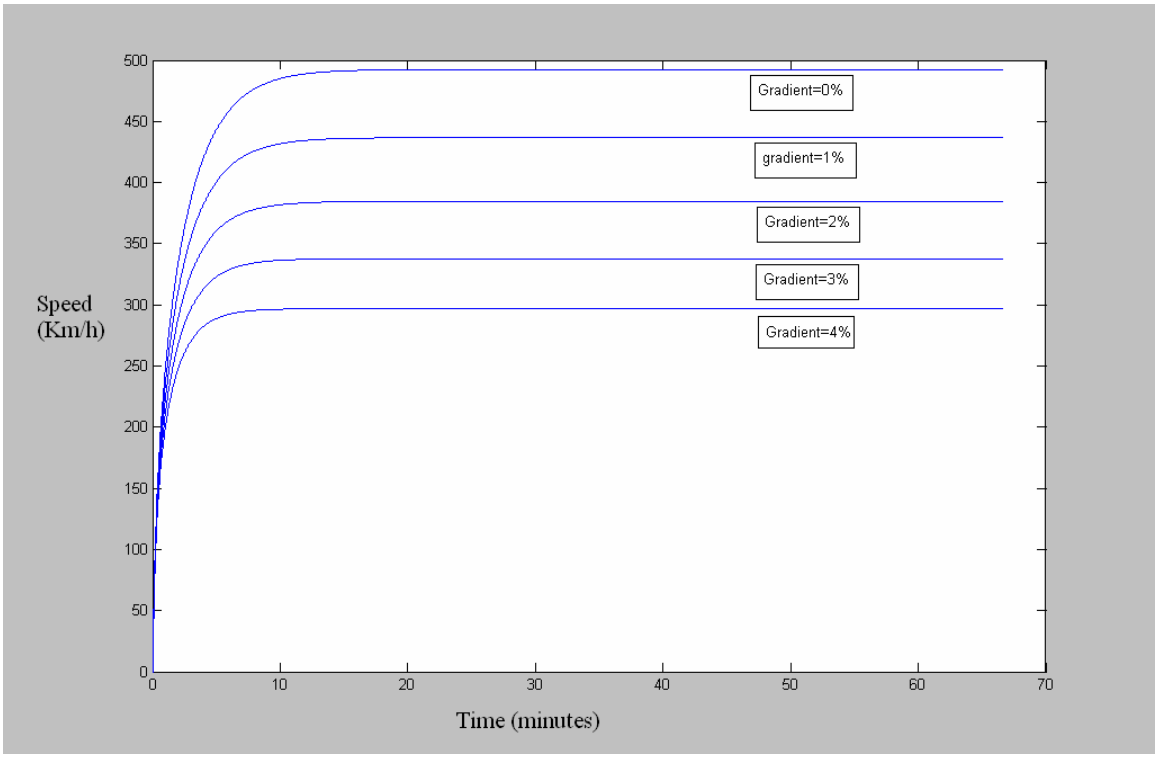


Figure 14: Speed v/s Time Curve for a 5000 hp Trainset for Different Gradients.

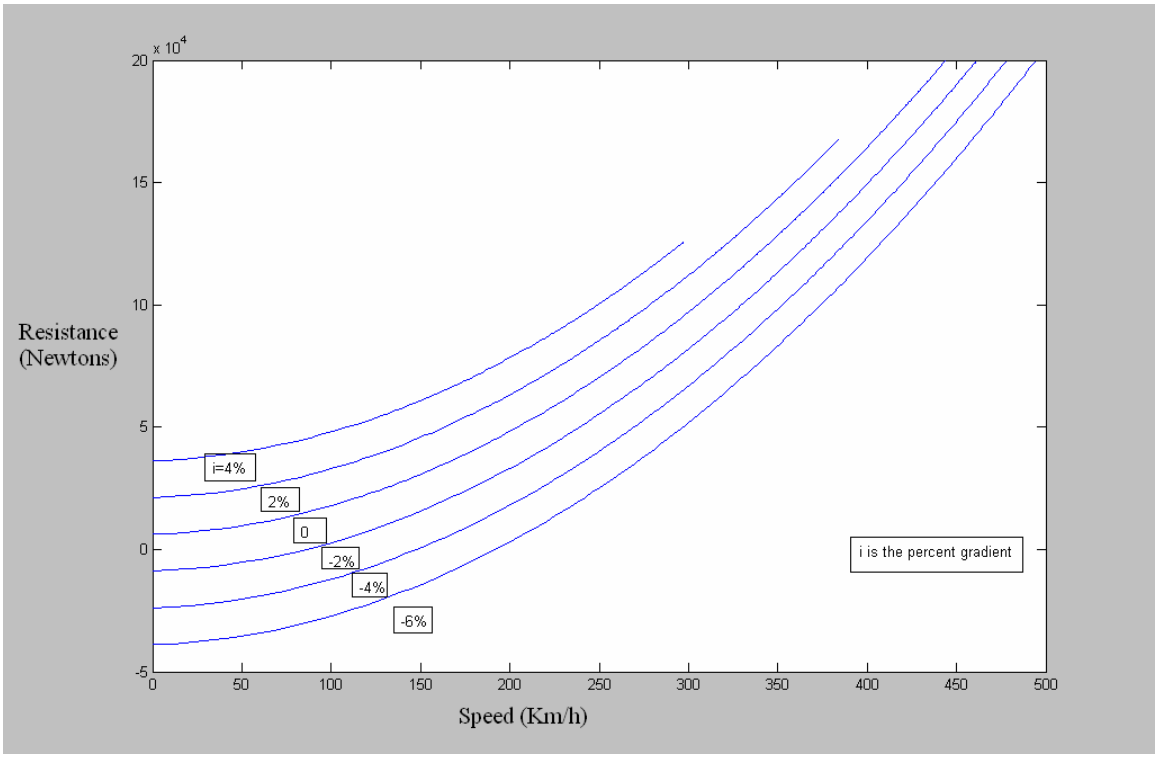


Figure 15: Vehicle Resistance for Various Gradient Values.

The available power these days can be as much as 16500 hp; If all the available power were to be applied all of a sudden, it would result in slippage and might prove harmful for the motor. A series of resistances is thus used to gradually apply the power available. Figure 16 shows the profile of Tractive effort with different values of applied power.

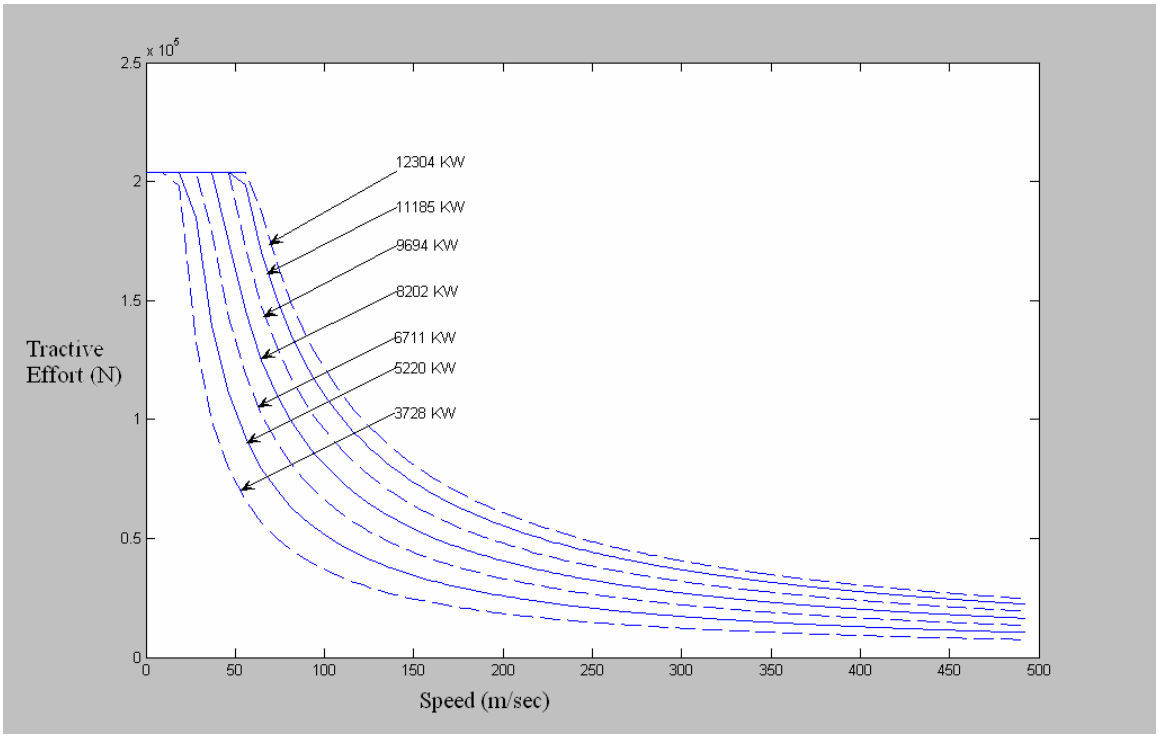


Figure 16: Tractive Effort v/s Speed at Various Levels of Applied Power.

The maximum value of speed that can be achieved by a vehicle is the speed at which the Tractive Effort and Resistance curves intersect. The maximum speed that can be achieved by a trainset with the above data can be ascertained by the plot shown in Figure 17.

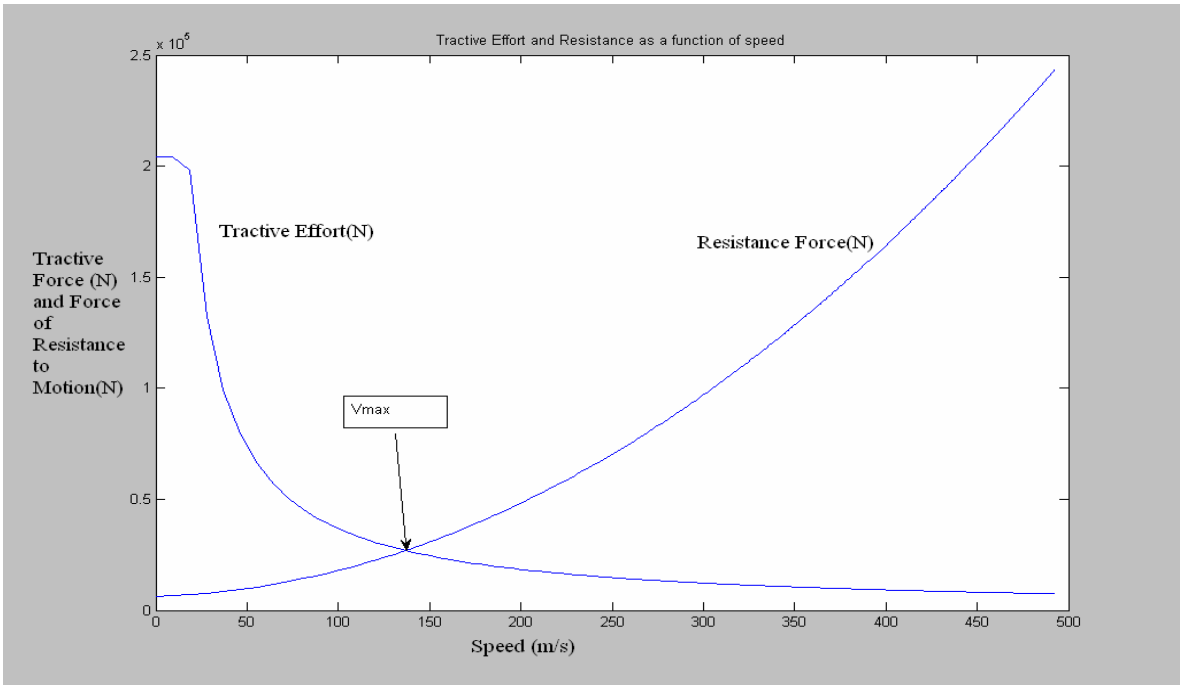


Figure 17: Tractive Effort and Force of Resistance to Motion as a Function of Speed.

The travel time estimation with such a dynamics based model can be accomplished using the Ode solver in MATLAB. The sub-routine can also be written to generate speed profiles for a particular using the given station to station distances. The profiles generated for this research haven been shown in Figure 18.

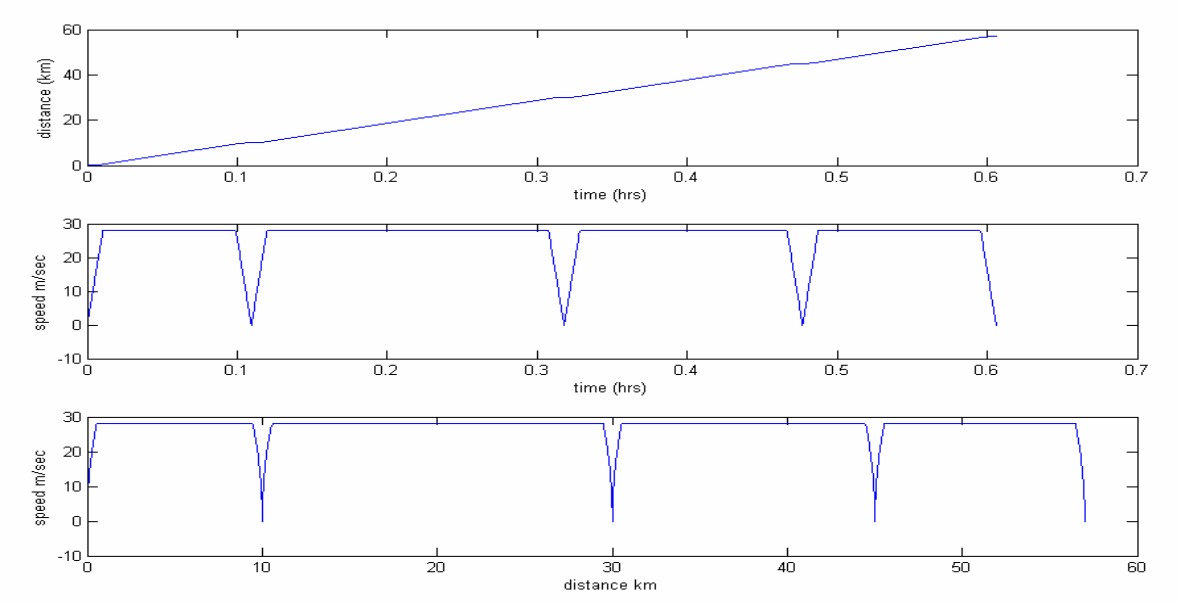


Figure 18: Speed Profile for a TGV Type Trainset on a Given Route.

4.3.1.2 Linear Acceleration Decay Model

In a linear acceleration decay model, the acceleration is assumed to decay linearly as a function of speed. Thus, the equation for acceleration as a function of speed at a time t is given by,

$$\frac{du_t}{dt} = \alpha - \beta u_t \quad \text{..... 13}$$

Where,

The LHS of the equation represents acceleration

u_t is velocity in mph at time t seconds

α and β are constants

Assuming the train to start from rest, and integrating the above equation within the limits $(0, u_t)$ the following expression is obtained for the speed of the train at time t,

$$u_t = \frac{\alpha}{\beta}(1 - e^{-\beta t}) \text{ ..... 14}$$

The coefficients α and β can be found if the speed time values for a trainset are known. This process has been explained later in the document.

The dynamics model requires a lot of data as input to generate meaningful results, for example a user should know details of the trainset performance as well as alignment details to be able to effectively use the model. Since, such detailed data was not available for this research a linear acceleration decay model was used for this study. The coefficients for the linear decay model were obtained by using technology based speed-time data published by the U.S. DOT.

4.3.2 Line-haul Time

Line haul time is that part of the travel time during which the passenger is actually traveling in the mode. Normally line haul distance will be the longest part of the journey but actual line haul time is relatively much less if time-distance proportions are considered. This becomes truer and much more apparent in faster modes of transportation e.g. Air and Maglev. Line-haul time is further composed of the following components.

4.3.2.1 Acceleration Time

As mentioned earlier, linear decay model was assumed for calculation of acceleration time. The following relationship was assumed,

$$\frac{du_t}{dt} = \alpha - \beta u_t \text{ ..... (1)}$$

Where,

The LHS of the equation represents acceleration

u_t is velocity in mph at time t seconds

α and β are constants

Assuming the train to start from rest, and integrating the above equation within the limits (0, u_t) the following expression was obtained for the speed of the train at time t,

$$u_t = \frac{\alpha}{\beta}(1 - e^{-\beta t}) \quad \text{..... (2)}$$

Using relationship (2), the time required by the train to a particular cruise velocity can be obtained. Performance values (speed-time) for different technologies were used to find the decay constants. These speed-time values have been documented in High-Speed Ground Transportation for America. (September 1997), A Report by U.S. Department of transportation and Federal Railroad Administration. The values of the constants have been listed in Table 3;

Table 3: Coefficients for Different Types of HSGT Technology Options.

Technology Type	α	β
Maglev	2.566	0.007
HSR	1.401	0.007
Accelerail-150	1.947	0.012
Accelerail-125	1.88	0.014
Accelerail-110	1.254	0.01
Accelerail-90	0.853	0.008

4.3.2.2 Deceleration Time

Deceleration time is the time required by the train to come to rest from a given speed. It was calculated after obtaining acceleration time. For calculation of deceleration distance, a constant deceleration of 0.75 m/s^2 was assumed.

4.3.2.3 Cruise Time

The amount of time the train travels at constant speed in the cruising stage is cruise time. For the calculation of cruise time, the total cruise distance was first calculated by subtracting the sum of distance required for acceleration and distance required for deceleration from the total distance to be traveled. Then cruise time was obtained by dividing the Cruise distance by the cruise speed.

The actual code for travel time calculation can be found in Appendix B. The line-haul time code is a function that has distance (in miles) as the input argument and returns

travel time in seconds. The relevant check for ability/inability of the train to reach maximum velocity in case of insufficient distance is included in the code. If the distance between stations was found insufficient for the three regime profile, the acceleration and deceleration were assumed constant at 0.75 m/sec^2; and the time was calculated by assuming only two stages to the journey viz. acceleration and deceleration.

In order to achieve faster computation speed, the line haul time was calculated for every connected link in the network and stored in a matrix that was directly operated upon instead of performing the travel time calculation for each route on the fly.

4.3.3 Access and Egress Time

Access and egress times were calculated as drive times to and from county centroids from the stations using map point that is added as a plug-in in the Virginia Tech TSAM. An average processing time of 20 after access and 10 minutes before egress was also added as a part of this step.

4.3.4 Dwell Time

Total dwell time is based on the number of stops made by the train excluding the starting and ending points and any transfers or layovers in case of connecting trains. An average dwell time value of 3 minutes was assumed, but it is not hard coded and is an attribute assigned to each route structure, it can therefore be modified by the user. Dwell time is added to the link travel time (recall line-haul time) after calculation of the line haul time for the link.

4.3.5 Transfer Time/Layover Time

Since time based schedules for the rail technology types being evaluated (it is beyond the scope of this study to formulate schedules) were not available. It was not possible to calculate the layover or transfer time directly in cases where a connection is needed. Each leg of the journey was considered as a new journey and the schedule delay (explained later) was calculated for each leg. Schedule delay is a function of frequency and can be thought of as the average layover a passenger is likely to undergo when taking a connecting train. The network building algorithm developed in this research was implemented as a preprocessor to Floyd's algorithm and was implemented when

establishing the connectivity of the nodes in the network. The dummy links inserted to connect the graph accounts for the layover a passenger is expected to encounter when transferring from one train to another.

4.3.6 Schedule Delay

Schedule delay {Teodorovic, (1988)} is a concept that comes into play when public transportation is considered. Since public transportation is available for use only at certain fixed intervals of time or as per schedule, i.e. the passenger may not be able to travel exactly at the time that he/she wants, but must take into consideration the frequency and timing of the service he/she wants to use and must adjust his or her travel plan accordingly. This introduces an inherent delay in the travel time of a passenger. The expression for such a delay {Teodorovic, (1988)} is given below;

$$S_d = \frac{T}{4 * N} \quad \text{..... 15}$$

Where,

S_d = Average Schedule Delay (hours)

T = Time for which the facility functions (hours)

N = Frequency of Departure

4.4 Travel Cost Estimation

The travel cost model developed in this research is crude but is useful nonetheless. A curve for travel cost per seat-mile was fit as a function of distance. The model gives cost of travel or fare from an origin to destination based on the distance between the same. Existing one-way fare data for Amtrak's Acela express service in the northeast corridor was used to obtain a base curve. Fitting was done with the help of a software package called 'Curve Expert'. A Harris model was obtained. Coefficient of correlation was given by the software as 0.9384.

Equation (4) is the general equation for the model.

$$y = 1 \Big/ a + bx^c \quad \text{..... 16}$$

Where,

a, b, c are calibrated constants

x is the distance in miles

The calibrated constants are listed in {Table 4}.

Table 4: Calibrated Constants for the Harris Model.

a	-0.36479
b	0.328033
c	0.333028

The base curve from Acela express data was extended to obtain values of fare up to 4000 miles. The average seat mile cost for this data was found to be 0.211 $/per-seat mile. It was found that this average value corresponded to a distance of 408 miles.

A report by USDOT and FRA contains a detailed cost analysis for some modes of HSGT technology. The costs per seat mile for operation and maintenance (average for all corridors) and fare yield for each alternative technology type given in the study for year 2020 were added and discounted at a rate of 4% to reflect the present value. These seat-mile costs were then adjusted using a multiplication factor for each type of technology in order to obtain a family of curves. The details of these values are listed below in the Table 5.

Table 5: Cost Scaling Factors for Different HSGT Options.

Technology Type	Multiplication Factors	Cost per Seat-mile
Maglev	1.049	0.221
HSR	0.927	0.196
Accelerail-150	0.783	0.165
Accelerail-125	0.822	0.173
Accelerail-110	0.747	0.158
Accelerail-90	0.733	0.155

The family of curves obtained for different HSGT options is shown in Figure 19.

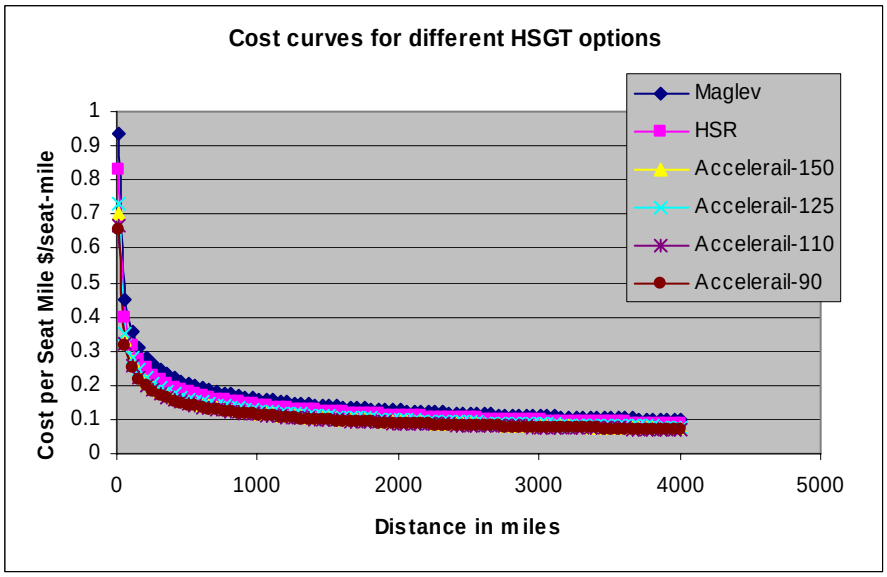


Figure 19: Cost curves for different HSGT Options.

The equations for these curves were coded as functions in MATLAB. The function accepts values as either a matrix or as single value of distance in miles to return total two-way travel cost. The return travel cost is simply two times the one-way cost, since that is the way Amtrak currently prices its tickets.

4.5 Integration of Travel Time and Travel Cost Model into the TSAM Model

The travel time and travel cost models formulated were integrated into TSAM model as lookup table functions used by the TSAM logit model. The schematic diagram of the integration process has been shown in Figure 20.

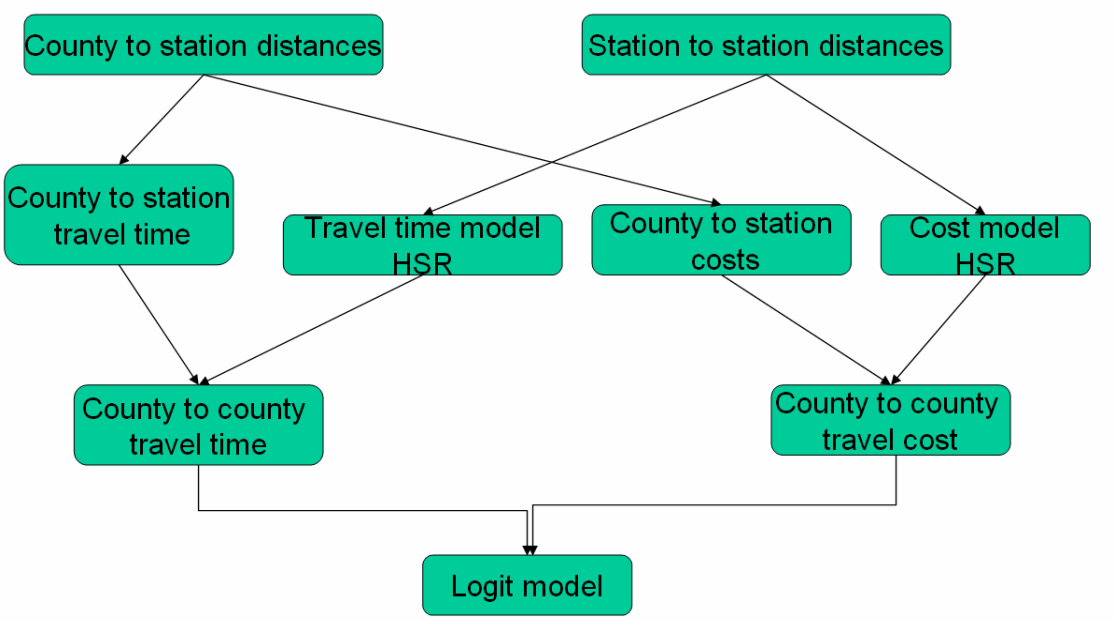


Figure 20: Linking the Travel Time and Travel Cost Model With TSAM Model.

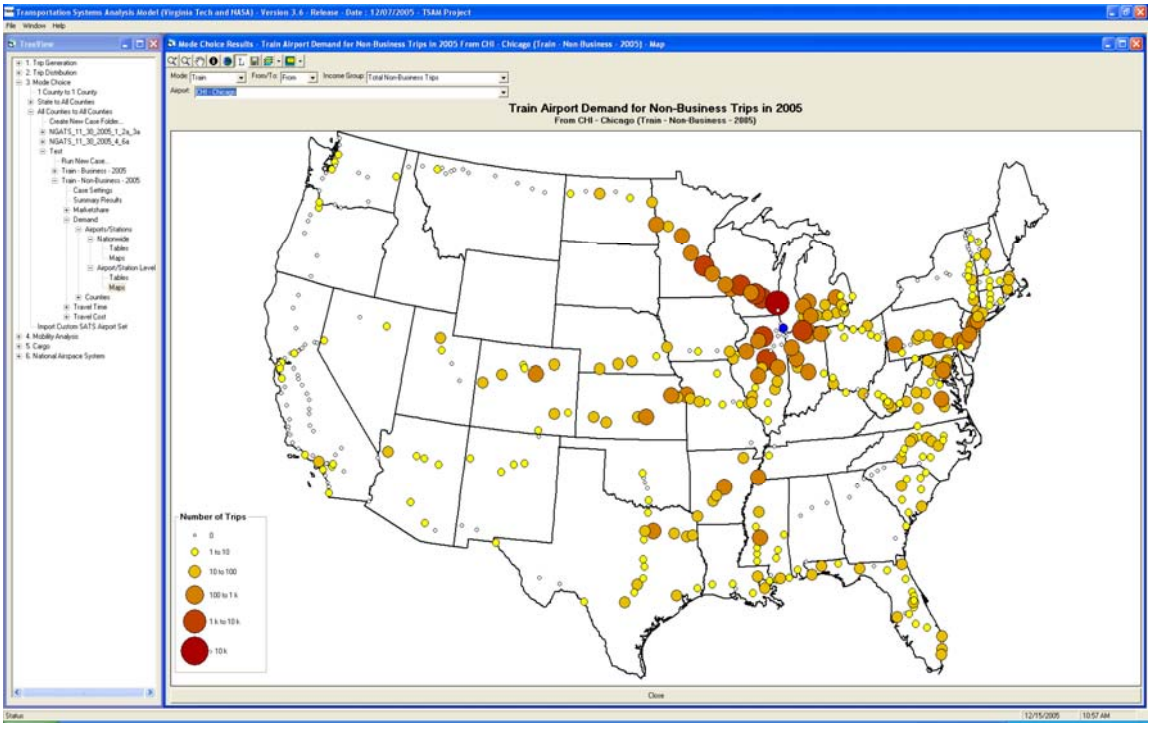


Figure 21: Screen Capture of the Rail Mode Integrated into the TSAM Model (Trips from Chicago to All- Non-Business).

5 Validation of Results

Validation of results was carried out by using a step by step approach. Having a systematic validation approach ensured that the results were consistent with expected trends and that there were no bugs in the MATLAB programs. The validation procedure used in this study is explained in this chapter.

5.1 Validation of Connectivity

The connectivity resulting from the algorithm was examined by obtaining a connectivity plot (Figure 23) in MATLAB and comparison with the existing Amtrak rail network (Figure 22). The main purpose was to check for any obvious problems such as missing links. Since the connectivity was derived from existing Amtrak schedules, it was expected to look similar to the network given by Amtrak on their official website. The obtained network and the actual Amtrak network are shown in Figure 22 and Figure 23.

Figure 22: Amtrak Network Published by Amtrak in 2004.

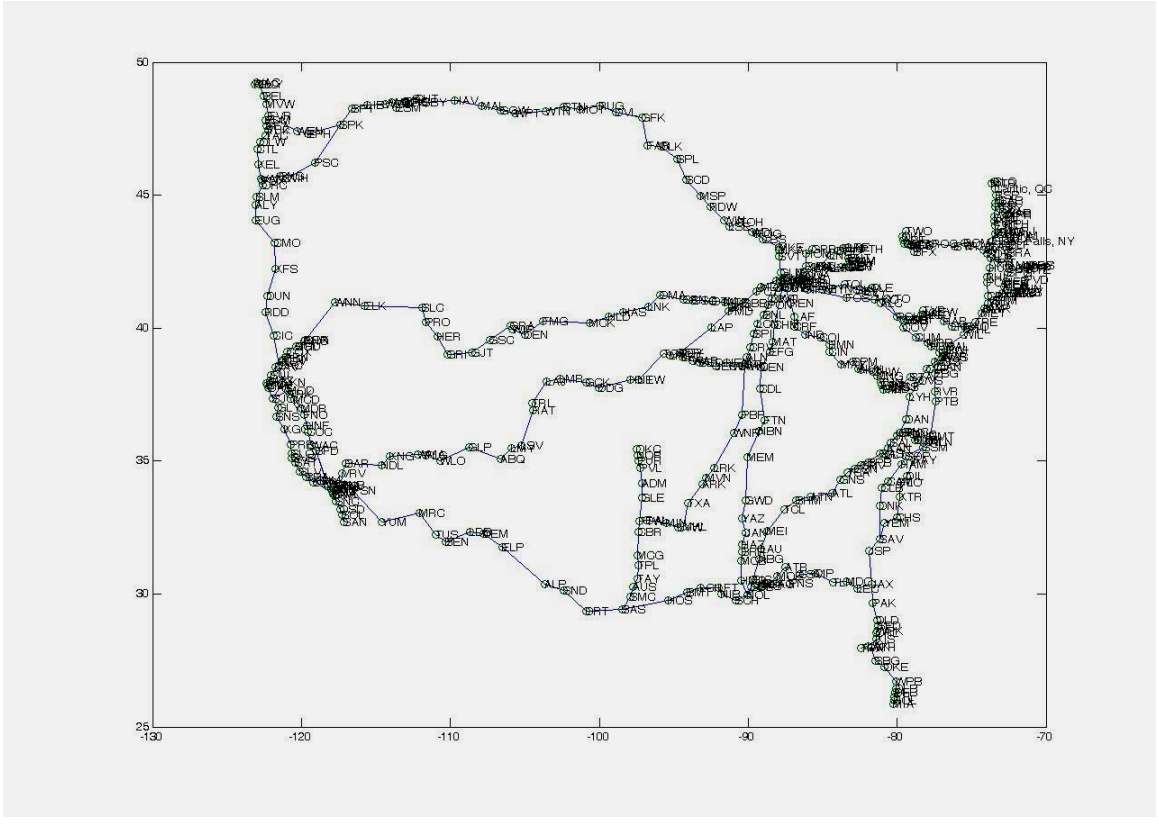


Figure 23: Network Obtained by Algorithm.

5.2 Verification of Performance Functions

In order to verify if the performance functions were returning correct values, a base connectivity was generated and the shortest path algorithm was run using these values without schedule delay, layover time and dwell time. An increase in travel time was expected with increasing distance. It was also expected that travel time values drop with improvement to higher end/faster technology types. The trends from Washington, DC and Chicago for Acela-90 are shown in the Figures 17 and 18. Figures for other technologies can be found in Appendix C. The observed trends were consistent with the expected trends.

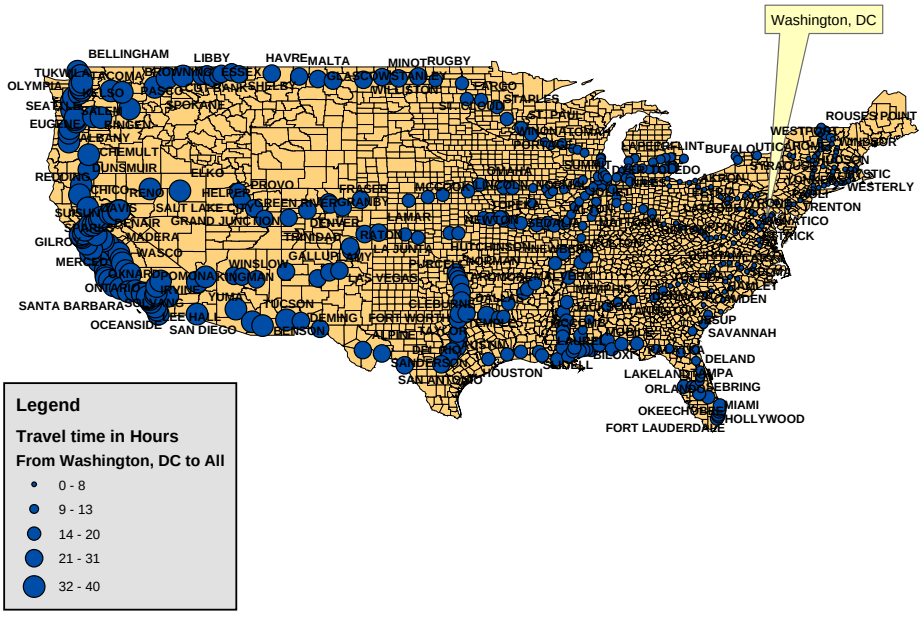


Figure 24: Acela-90, DC to All.

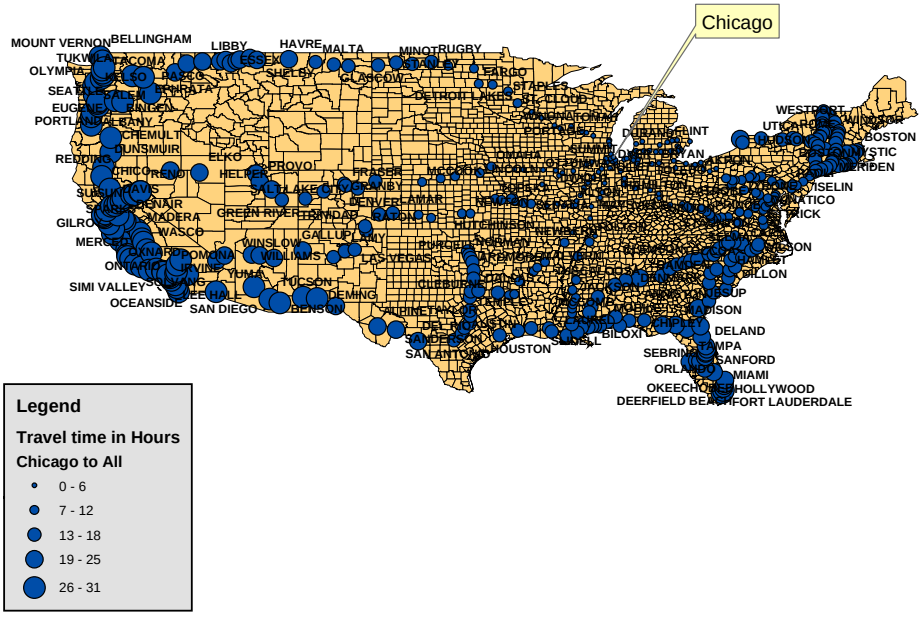


Figure 25: Acela-90, Chicago to All.

5.3 Validation of Travel Time

5.3.1 Verification against Existing Travel Time Data

Travel time values obtained using the calibrated train performance functions were compared with actual Acela Express travel time values in the northeast corridor. In order to perform the verification, the great circle distances between the stations in the northeast corridor were calculated using the station latitude and longitude and multiplied by a detour factor of 1.2 to account for curvature etc. The performance function for Acelarail-150 was used assuming a dwell time of 3 minutes. Accelerail-150 was selected for the comparison since it best represents the existing Acela service trainsets. The results of the comparison have been presented in Table 6 and Figure 26.

Table 6: Comparative Travel Times from the Model and Actual Travel Times from the Amtrak Schedules.

Travel Time from Washington, DC Union Station (in Minutes)		
Destination	Model Estimated	Actual Travel Time From Schedule
BAL	34.037	35
WIL	75.341	76
PHI	99.007	96
NWK	162.9	154
NYP	174.02	173
STM	216.47	226
NHV	249.99	269
PVD	323.72	355
BOS	359.84	396

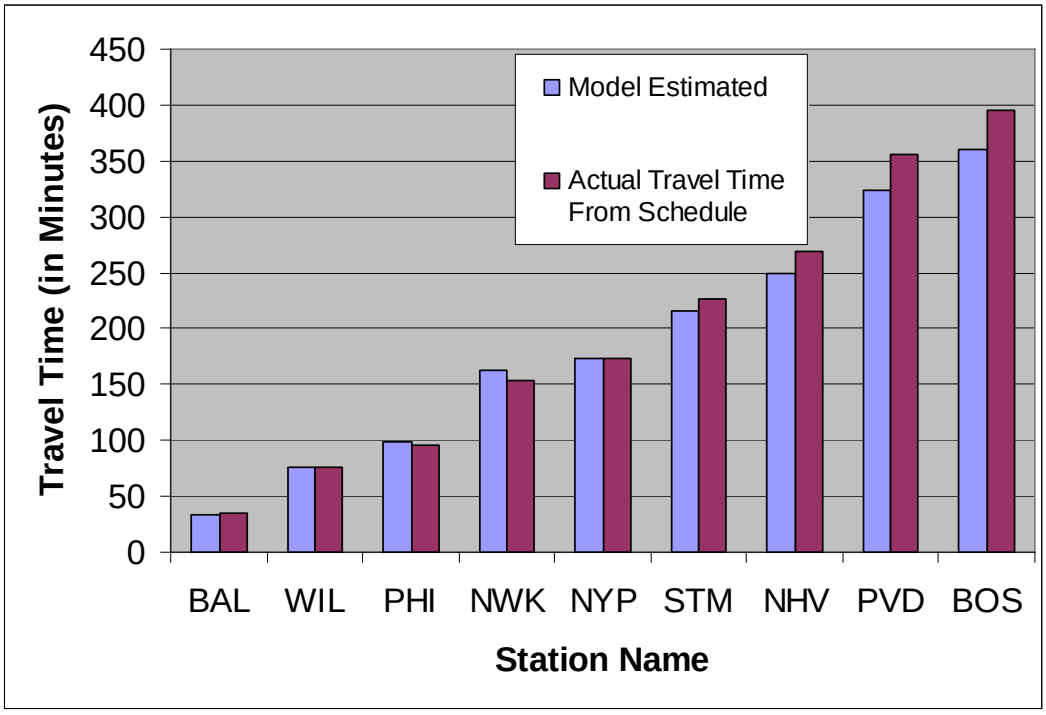


Figure 26: Travel Time Comparison for the Northeast Corridor.

It can be seen from the above plots that the travel times obtained from the model are fairly close to actual travel times. A larger discrepancy is seen in the last few values and is most likely the result of a discrepancy in dwell time values. This shows that more detailed information about the network characteristics could greatly improve the travel time estimation.

5.3.2 Verification of Travel Times for the Entire Network

Verification of travel times evaluated by the model was carried out by plotting travel time against distance, plotting speeds against distance and finally plotting a histogram to ascertain distribution of speeds. The expected trend for travel time was obviously, higher travel time for greater distances with the exception of transfers where a direct train would require a lower travel time as compared to a journey involving a transfer for the same distance. The expected results for speed were that no speed value be above the maximum speed allowable for a particular technology type. For example a speed of higher than 90 miles per hour for Acela-90 would be an indication of a problem. It was also necessary to ensure that speeds were not very low, some cases were found to have very low speeds, and one such case has been explained in the next chapter where the results are discussed. The plots for distance v/s time; distance v/s speed and histogram of speeds for Acela-90 are shown in Figures 19-21. Plots for other technology types are included in Appendix C.

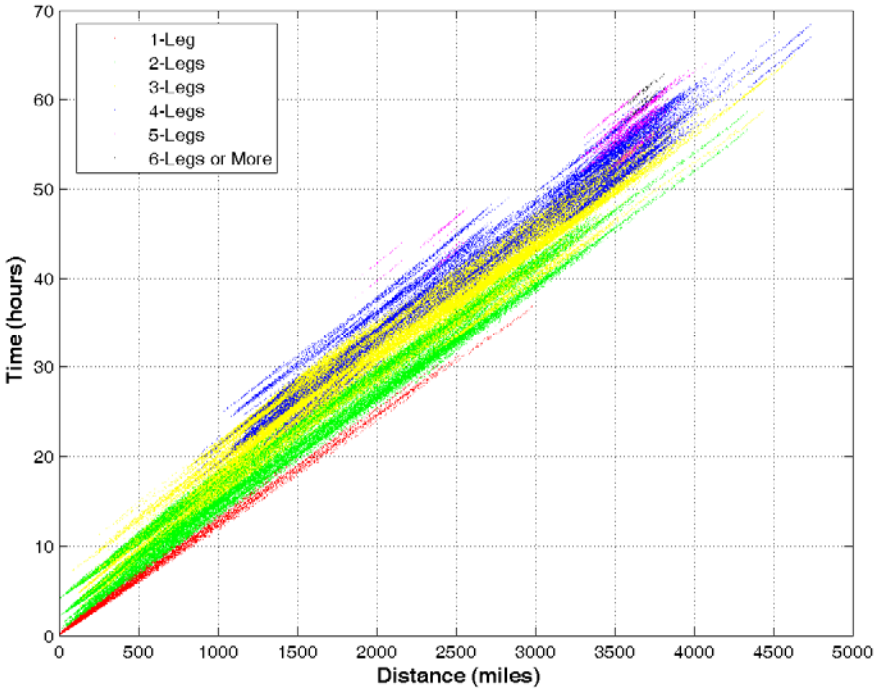


Figure 27: Time v/s Distance Plot for Acela 90.

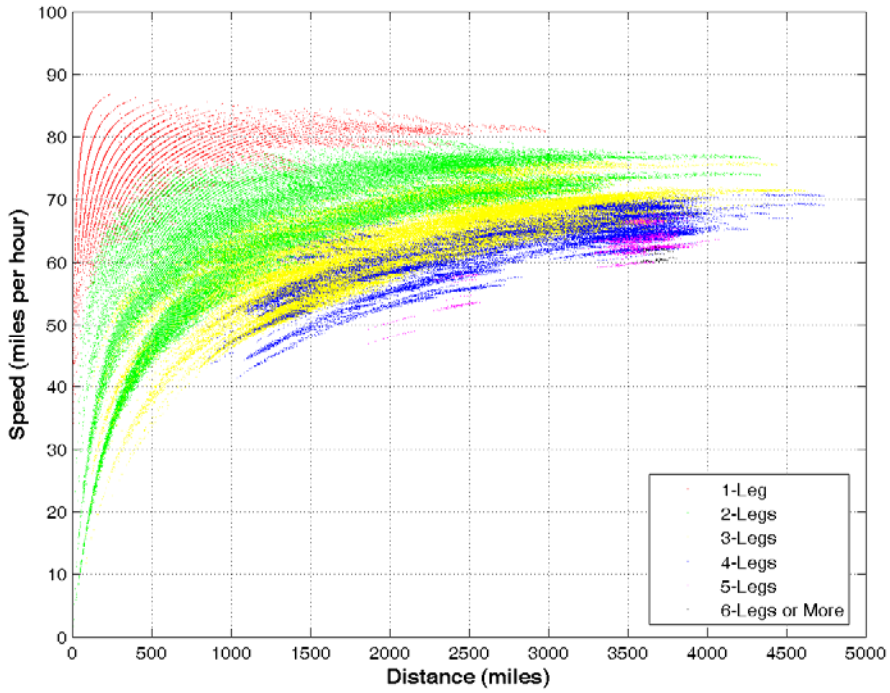


Figure 28: Speed v/s Distance Curves for Acela 90.

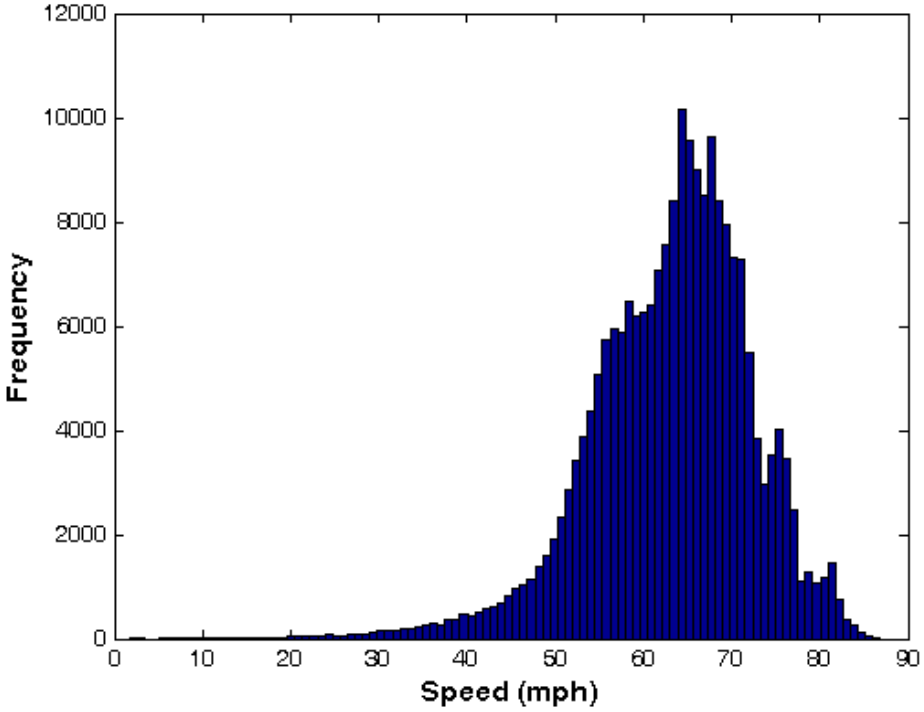


Figure 29: Distribution of Speeds for Acela 90.

5.4 Verification of Distances

The measure used for validation of distances is detour factor; it is obtained by dividing Obtained distance by the Great Circle distance between two points. Since the great circle distance is the shortest possible distance between two points, the detour factor can not be less than one. The histograms for the detour factors have been presented next. It was seen that some station pairs had a very high value of detour factor; the reason for the same is discussed in the next chapter where a specific case is considered and explained.

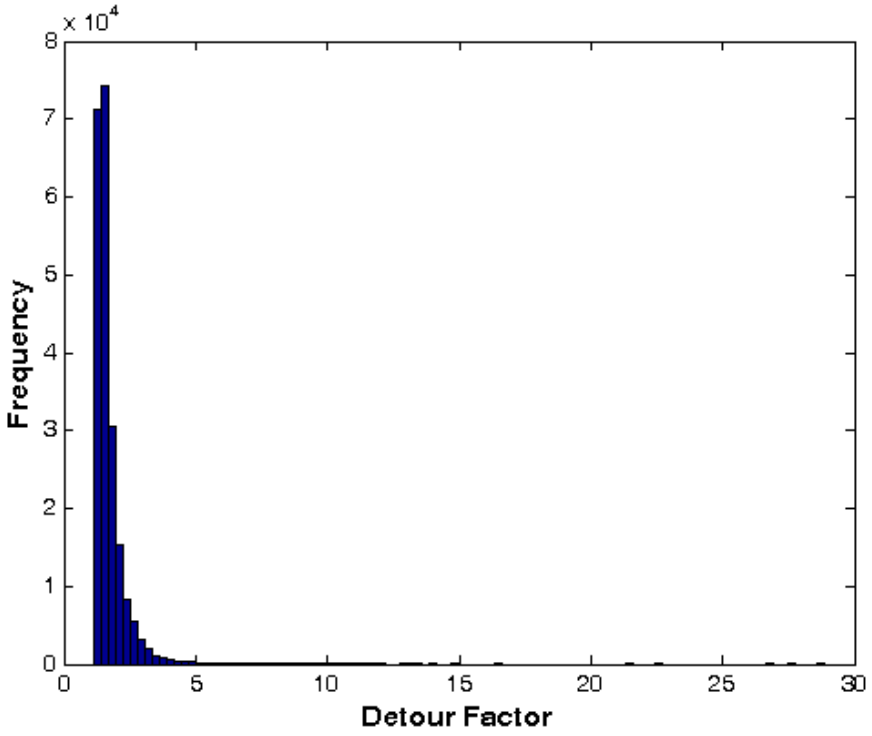


Figure 30: Distribution of Detour Factors for Acela 90.

After performing the checks on detour factors it was confirmed that the distances obtained were ok. To ascertain if the cost function was working correctly, distance was plotted against the travel cost. The expected trend was a monotonic increase in cost with the distance and higher cost for higher end technologies.

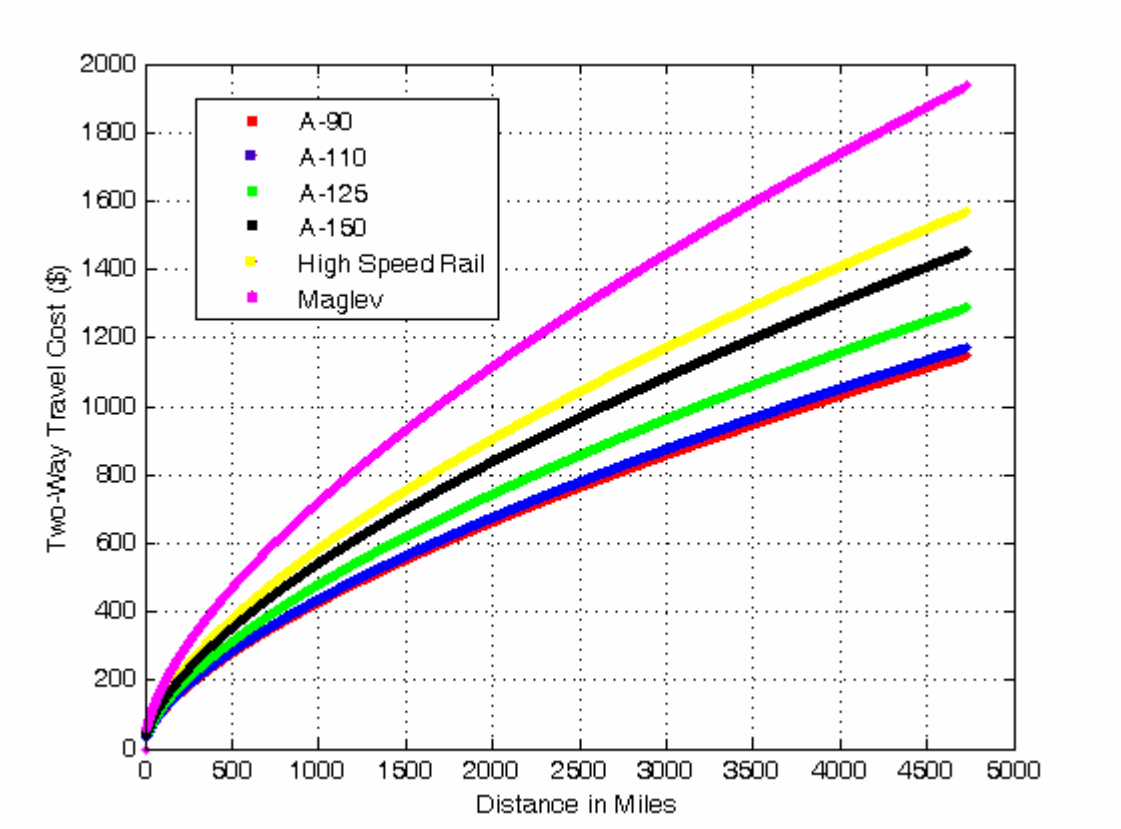


Figure 31: Travel Cost Trend for Different Technology Types.

The plots for distance v/s cost indicate an increasing value of travel cost as the technology type becomes faster; this is consistent with the expected trend.

6 Discussion of Results

This chapter is aimed at discussing the findings of this research. The chapter is divided into two parts; in section 6.1 a case study for travel time observations of some key corridors of the US Rail network is presented. Some seemingly anomalous cases like high detour factors and low speeds mentioned in the previous section have been discussed in section 6.2.

6.1 Case study for Key Corridors of the Amtrak Rail Network

The methodology developed in this research effort was tested on the Amtrak rail network, in order to ascertain whether the method gives rational values of travel time it was necessary that some comparisons be made with real data. The graphs presented below show how the total travel time estimates (including schedule delay) obtained using the model compare with the actual travel time values published with Amtrak. Even though the technology types for Amtrak and those modeled are different, Amtrak travel-times are used as a base measure in order to make sure that the values obtained by the model are not excessively over or under estimated.

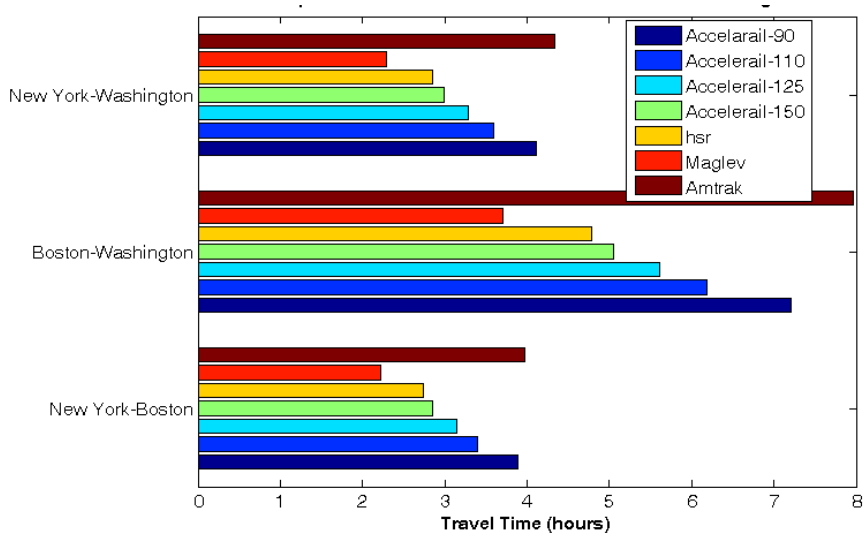


Figure 32: Comparative Travel Times for Different Rail Technology Types in the Northeast Corridor.

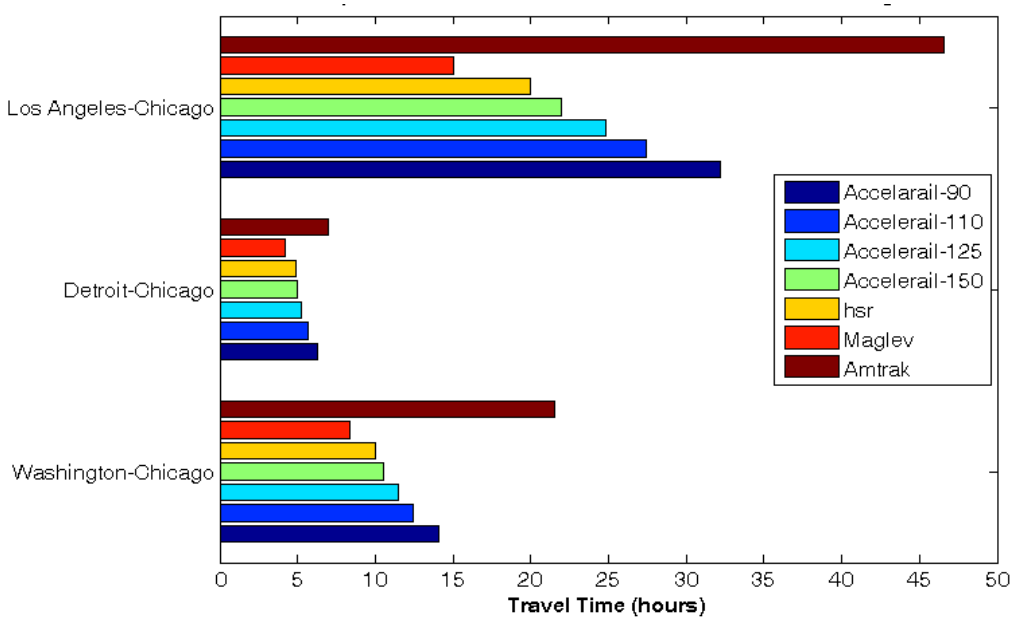


Figure 33: Comparative Travel Times for Different Rail Technology Types in the Midwest or Chicago Hub Network.

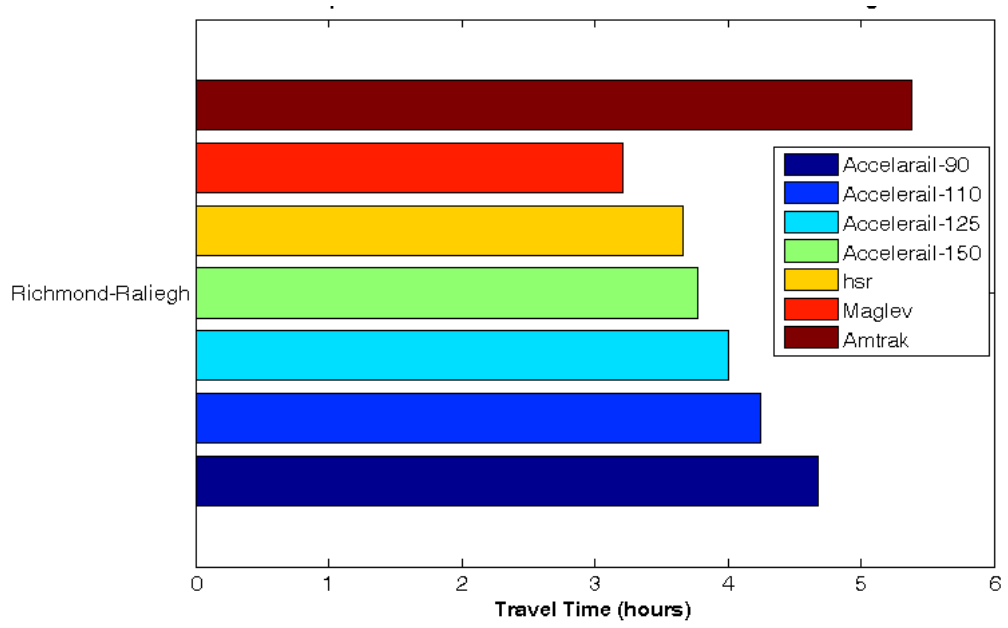


Figure 34: Comparative Travel Times for Different Rail Technology Types in the Richmond Raleigh Corridor.

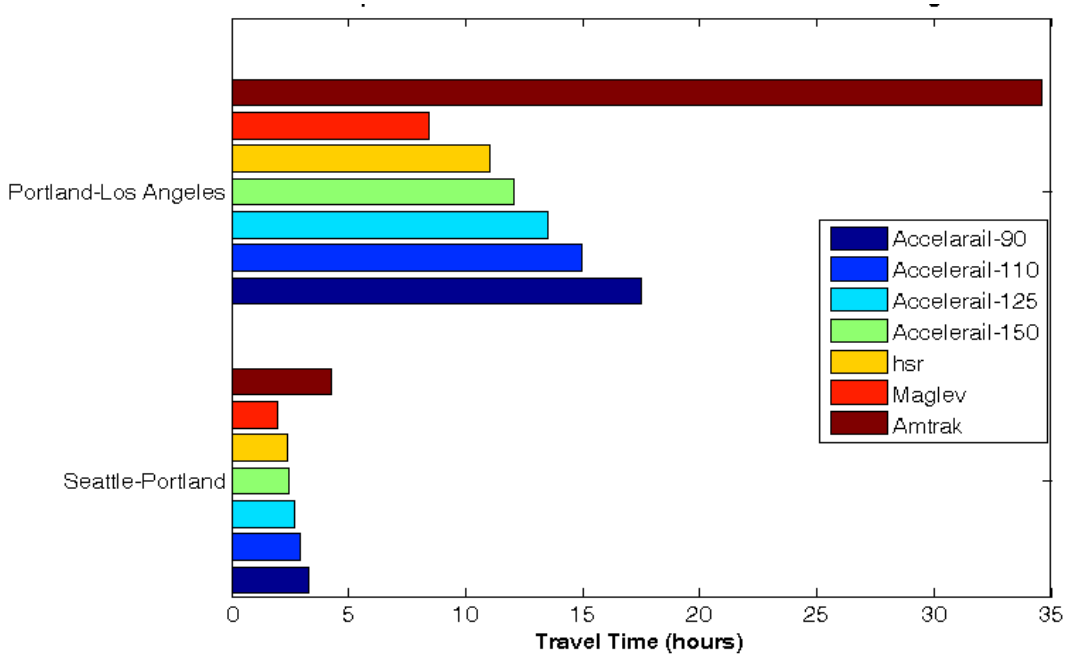


Figure 35: Comparative Travel Times for Different Rail Technology Types in the Pacific Northwest and California.

From the above plots it can be clearly seen that for shorter distance trips, the total travel time does not reduce significantly with an upgradation in technology type, but travel time significantly reduces with an upgradation of technology type in the higher distance corridors. This observation leads to the conclusion that for shorter distance corridors (distances < 600 miles) frequency of service is a major governing factor for travel time improvement. Thus, it can be said that a good operational efficiency can be achieved for such corridors if the scheduling and management of rolling stock and timetabling is optimized as in the airline industry. This will allow the rail company to provide better level of service with the existing inventory rather than making heavy investments in acquisition of higher end rolling stock.

6.2 Explanation of Anomalous Results

Some values of block speeds and detour factors seemed to be out of the expected range. The reasons for such results are explained in this section.

6.2.1 Low Block Speed

It was observed that some block speeds were very low, to the order of approximately 4 miles per hour. The example discussed explains such situations. In order to go from Coliseum Airport to Berkeley, a passenger is required to change trains at Emeryville where there is a layover time of 4 hours. As a consequence, the travel time becomes very high and the block speed drops down to a very low value. Figure 36 is a diagrammatic representation of the situation.

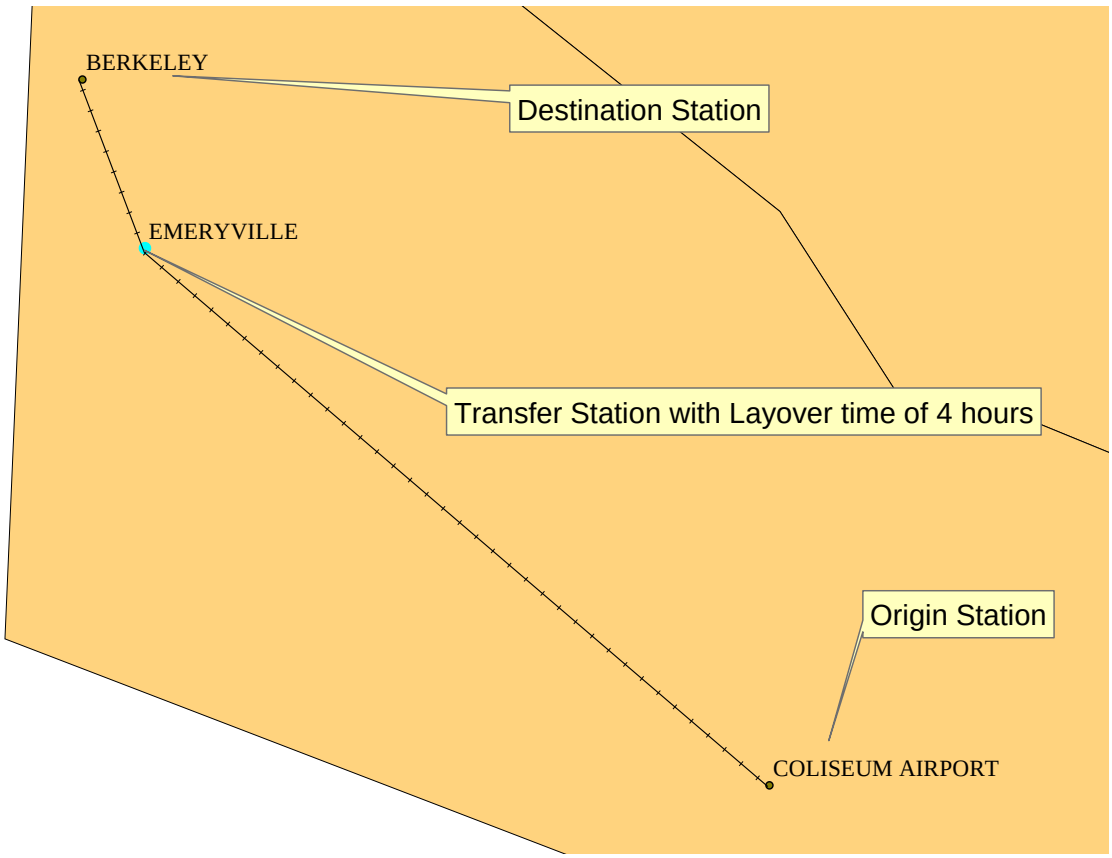


Figure 36: Explanation for Extremely Low Values of Speed.

6.2.2 Excessively High Detour Factors

Another key observation was that the detour factor for some station pairs was abnormally high. The cause for this was that even though some stations are in close proximity to one another, there is no rail service between them. The result is that the traveler is required to take a long way around and this translates into a large detour factor. Consider an example

where a passenger traveling from Rouses Point, NY to St. Albans, VT. Even though the stations are not far apart spatially, using rail to go from station to the other involves a huge detour, as a result the distance required to be traveled by the passenger is much greater than the great circle distance. Figure 37 is a diagrammatic presentation of the case explained above.

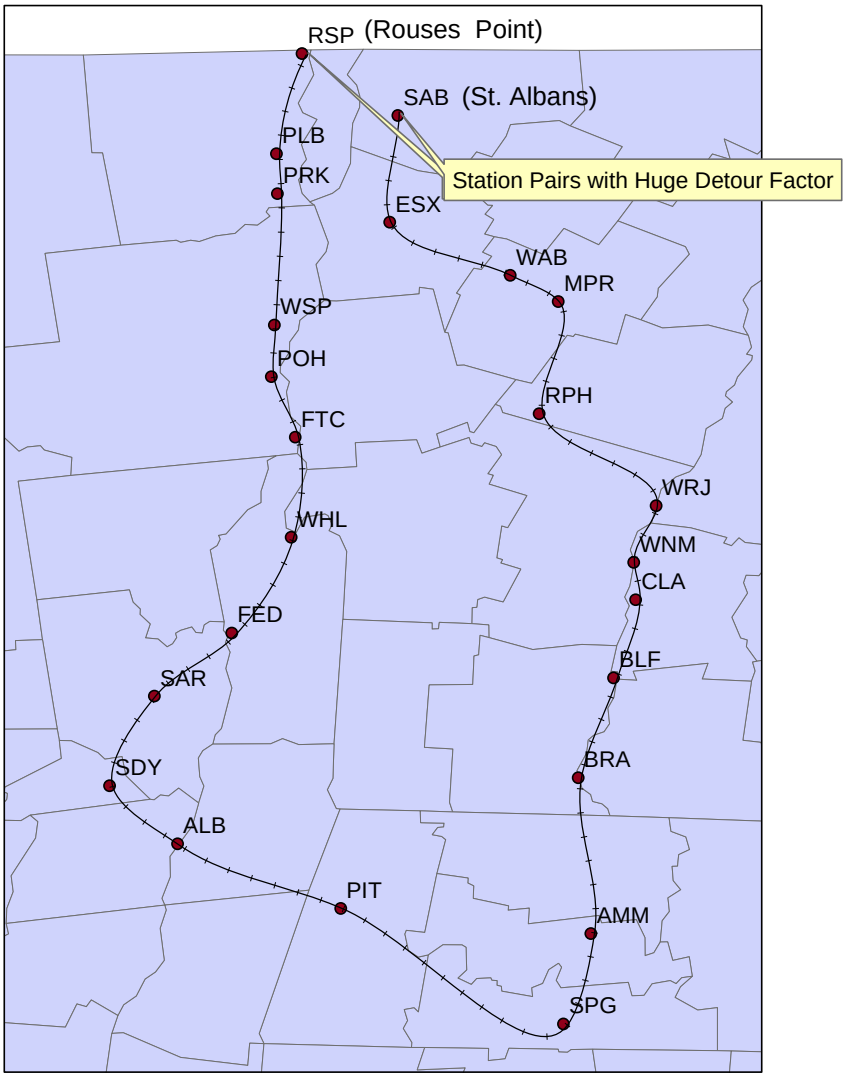


Figure 37: Explanation for High Detour Factor.

6.2.3 Travel Time Inconsistencies for Some Station Pairs

It was observed that the travel time values for some station pairs were higher than some station pairs that were relatively farther apart, when the schedule delay was subtracted from the total travel time. The reason for this is that the travel time minimization was carried out for total travel time rather than just line haul and transfer time, as a result some routes that would be the shortest without schedule delay were rejected and routes that had a lower total travel time were retained. For example, a route might seem to be attractive when schedule delay is not considered, but it is also less likely to be taken if it is offered only once a day. Since the total time is minimized, a route which has a higher layover and line-haul time between the same station pairs may get selected as the shortest path if it has lower schedule delay associated with it.

7 Conclusion

A credible and robust model for evaluation of Rail systems has been developed as a part of this research effort. The results from this model should be useful for purposes of mode choice modeling and the network topology generated will pave the way for performing network assignment should it be required in the future. As with any other model, this model also has some drawbacks which might limit its applicability in some conditions. Since it is known that the computational complexity of Floyd's Algorithm is $O(n^3)$(n being the number of nodes), using an exploded network will increase the runtime for the algorithm. Currently the core network of 464 nodes is exploded into a network containing 1230 nodes; the runtime for Floyd's algorithm for an all-to-all case for a network of this size is approximately 9 minutes on a Pentium IV 1.7 GHz machine with 512 MB of RAM. The all-to-all scenario for the mode choice module of TSAM currently takes 4 hours of 240 minutes to run. The current runtime of the network algorithm though within bounds for this current scenario, will increase drastically for a larger network of say 10,000 nodes. Even though the existing condition of the US rail network rules out any such drastic increase in the number of nodes, work will be needed to be carried out to bring the runtime of to a lower value if there were to be a major shift in mode choice behavior and the rail network becomes larger.

Recommendations for Future Work

Even though the model developed in this research is useful, additional improvements which could be made are listed below;

1) Addition of a Capacity Constraint on Frequency since a rail line can handle only a limited number of operations for given headway constraints.
2) Application of Dijkstra's Algorithm in order to calculate shortest path for one to one or one to all cases instead of using Floyd's algorithm. This will reduce runtime for such cases as it will eliminate unnecessary calculations.
3) Extension of the network built to perform assignment in the network (Recall, the last step in the four step approach).
4) Using multigraph algorithms to decrease the total runtime.

References:

1. Chris Barrett, Riko Jacob and Madhav Marathe (May 1998) Formal Language Constrained Path Problems—Approved Public Release by Los Alamos National Laboratory, USA.

2. C.S. Papacostas, P.D. Prevedouros (2002) Transportation Engineering and Planning (IIIrd Edition)—Published by Prentice Hall (India).

3. David A. Hensher, Kenneth J. Button (2000) Handbook of Transport Modelling (Ist Edition)—Published by Pergamon, Elsevier Science Ltd.

4. David Levinson, Adib Kanafani and David Gillen (July1998) Air, High Speed Rail, or Highway-- A Cost Comparison in the California Corridor submitted to Transportation Quarterly.

5. David Levinson, David Gillen, Adib Kanafani, Jean-Michel Mathieu, (June 1996) The Full Cost of Intercity Transportation – A Comparison of High Speed Rail, Air and Highway Transportation in California, RESEARCH REPORT UCB-ITS-RR-96-3, Journal of Public Transportation, Volume 2, No. 4, p.19-39.

6. David Levinson, RAIL REINVENTED?--A Brief History of High Speed Ground Transportation.

7. Friedrich, M., Haupt, T., Nökel, K. (1999) Planning and Analyzing Transit Networks –An Integrated Approach Regarding Requirements of Passengers and Operators.

8. GAO's Report to Congressional Committees (1998) INTERCITY PASSENGER RAIL--Financial Performance of Amtrak's Routes, p. 28-42.

9. High-Speed Ground Transportation for America. (September 1997), A Report by U.S. Department of transportation and Federal Railroad Administration.

10. Heinz Spiess, (June 1993) Transit Equilibrium Assignment Based on Optimal Strategies: An Implementation in EMME/2-- EMME/2 Support Center, Switzerland.

11. Mark Allen Weiss, (1992) Data Structures and Algorithm Analysis in C.

12. Md. Shoaib Chowdhury, Steven I-Jy Chien, (January 2001), OPTIMIZATION OF TRANSFER COORDINATION FOR INTERMODAL TRANSIT NETWORKS -- 80th Annual Meeting, Transportation Research Board, p. 7-11.

13. Myron Lewellen, Kerim Tumay (1998) Network Simulation of a Major Railroad-- Proceedings of the 1998 Winter Simulation Conference.

14. Nicholas J. Garber, Lester A. Hoel (2001) Traffic Engineering and Highway Engineering (IIIrd Edition)—Published by BROOKS/COLE.

15. Paul Martin, (1999) TRAIN PERFORMANCE AND SIMULATION-- Proceedings of the 1999 Winter Simulation Conference.

16. Peter Hall, Dan Leavitt, Erin Vaca, High Speed Trains for California—(Volume II: Detailed Segment Descriptions, Cost Estimates, and Travel Time Calculations) Working Paper, The University of California Transportation Center University of California, Berkeley.

17. Robert H. Leilich, (1998) Application of Simulation Models in Capacity Constrained Rail Corridors-- Proceedings of the 1998 Winter Simulation Conference.

18. Statewide Rail Transportation Assessment (September 2002) Revised Draft Report-- Prepared by the California Department of Transportation with Assistance from: Booz-Allen-Hamilton, Wilbur Smith Associates, System Metrics Group, p. 3-23.

19. Dusan Teodorovic, (1988) Airline Operations Research—Published by Gordon and Breach Science Publishers S.A.

20. Thomas Lindner, (2000) Train Schedule Optimization in Public Rail Transport—PhD Dissertation, University of Braunschweig (Germany).

21. Amtrak Corporation, (2004) Passenger Rail schedules available: http://www.amtrak.com

22. Bureau of Transportation Statistics, (2004) National Transportation Atlas Database (NTAD) 2005 CD available: **http://www.bts.gov/publications/north_american_transportation_atlas_data**

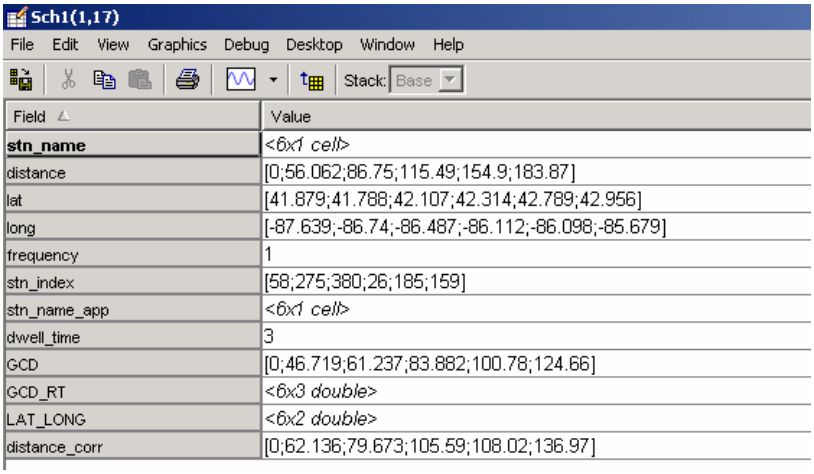
Appendix A:

The data structures and key files used in the programs are described in this section.

A.1 Definition of Key Data Structures and Files

Route Information Data Structure

The Route Information data structure is a struct array. This array contains information on various routes on the network. Shown in Figure A. 1 is a screenshot of the struct file used to store the route information. The field shown is a single route from a total of 39 routes.



Field	Value
stn_name	*<6x1 cell>*
distance	[0;56.062;86.75;115.49;154.9;183.87]
lat	[41.879;41.788;42.107;42.314;42.789;42.956]
long	[-87.639;-86.74;-86.487;-86.112;-86.098;-85.679]
frequency	1
stn_index	[58;275;380;26;185;159]
stn_name_app	*<6x1 cell>*
dwell_time	3
GCD	[0;46.719;61.237;83.882;100.78;124.66]
GCD_RT	*<6x3 double>*
LAT_LONG	*<6x2 double>*
distance_corr	[0;62.136;79.673;105.59;108.02;136.97]

Figure A. 1: Screenshot of the Fields Contained in the Struct File (some additional fields seen here were meant for verification purposes).

The key fields stored in the route information data structure are explained below:

1. **stn_name** (cell array of strings)

   This field contains the three letter IDs of all the stations in a route, the order of these IDs is consistent with the order in which the stops have to occur. These station IDs represent the station/transfer nodes.

2. **distance** (single column array)

   The distances with the origin as datum or 0 are stored in this field, thus the distance between any two adjacent stations is the absolute value of the difference between them (since we consider an undirected graph).

3. **LAT_LONG** (array with two columns)

   This field contains the latitude and longitude information of the stations, the file is useful for projection of the route on a map and also useful for carrying out checks for

detour factors or calculation of distances in case the distance between the stations is not known.

4. **stn_name_app** (cell array of strings)

   The route numbers are concatenated to the names of stations in this field, these nodes represent line nodes. No transfers can take place on line nodes. This data is used in building the network.

5. **dwell_time** (single value)

   Average dwell time values in minutes for each route is stored in this single value field, the dwell times are spilled over the links in this study.

6. **frequency** (single value)

   Frequency for the route is stored in this field, the average expected connecting time or layover is calculated using this value.

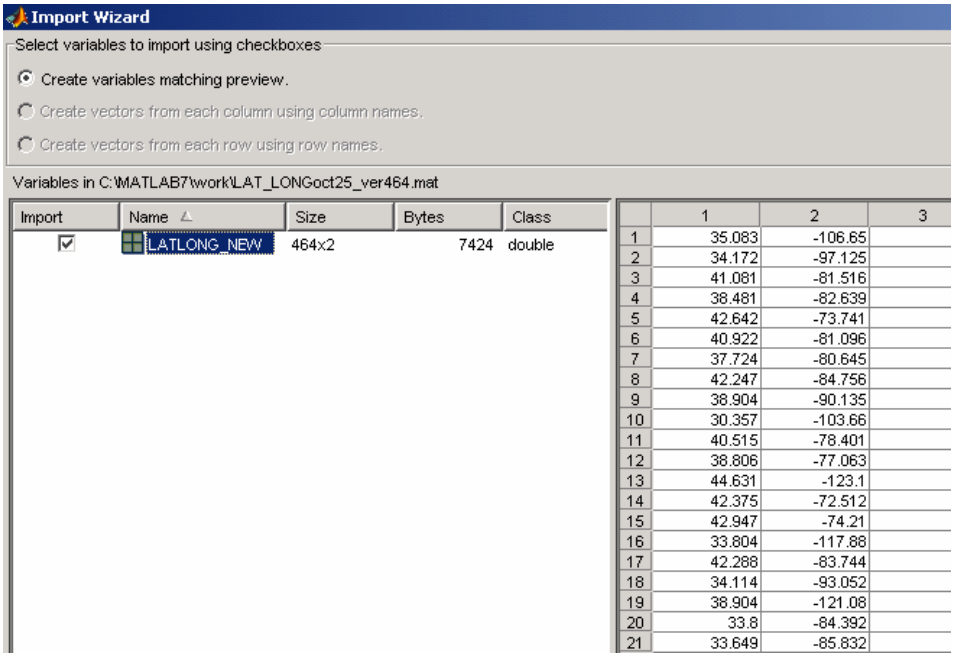
Data Files Used

1. **stationcodes**

   All the station IDs for which the analysis is being carried out are stored in this file, the station Ids are three lettered. This file is a cell array of strings (464 by 1) and is the main file on which the indices are based, the station Ids are sorted in alphabetical order.

2. **LATLONG_NEW**

   The latitude and longitudes for all the stations is stored in this file the length of the file is the same as that of stationcodes but it has 2 columns and all numeric values. This file is used in calculation of great circle distances between station pairs; the great circle distance is used as a validation tool for distances. A screenshot of this file has been shown in Figure A. 2.



	1	2	3
1	35.083	-106.65	
2	34.172	-97.125	
3	41.081	-81.516	
4	38.481	-82.639	
5	42.642	-73.741	
6	40.922	-81.096	
7	37.724	-80.645	
8	42.247	-84.756	
9	38.904	-90.135	
10	30.357	-103.66	
11	40.515	-78.401	
12	38.806	-77.063	
13	44.631	-123.1	
14	42.375	-72.512	
15	42.947	-74.21	
16	33.804	-117.88	
17	42.288	-83.744	
18	34.114	-93.052	
19	38.904	-121.08	
20	33.8	-84.392	
21	33.649	-85.832	

Figure A. 2: Screenshot of LATLONG Data File.

3. **Precedence Matrix P**

This file stores the shortest paths in an efficient flat file having preceding node information, the variable P is declared as a global variable. A function uses the precedence matrix to display the shortest path. A screenshot of the file has been shown in Figure A. 3

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	687	723	748	799	885	741	1212	862	671	717	988	581	776	839	603	
2	525	2	723	748	799	885	741	1212	649	671	717	988	581	776	839	603	
3	525	687	3	748	799	885	741	1212	649	671	717	988	914	776	839	603	
4	525	687	723	4	799	885	741	1212	649	671	717	734	914	776	839	603	
5	525	687	723	748	5	697	741	1212	649	493	717	988	914	776	839	603	
6	525	687	723	748	799	6	741	1212	649	671	717	988	914	776	839	603	
7	525	687	723	748	799	697	7	1212	649	671	717	734	914	776	839	603	
8	525	687	723	748	799	885	741	8	649	671	717	988	914	776	839	603	
9	525	687	723	748	799	885	741	1212	9	671	717	988	914	776	839	603	
10	525	687	723	748	799	885	741	1212	649	10	717	617	581	776	839	603	
11	525	687	723	748	799	885	741	1212	649	671	11	988	914	776	839	603	
12	525	687	723	748	799	697	741	1212	649	493	717	12	914	776	839	603	
13	525	687	723	748	799	885	741	1212	649	671	717	988	13	776	839	603	
14	525	687	723	748	1112	697	741	1212	649	493	717	988	914	14	839	603	
15	525	687	723	748	837	697	741	1212	649	493	717	988	914	776	15	603	
16	525	687	723	748	799	885	741	1212	862	671	717	988	581	776	839	16	
17	525	687	723	748	799	885	741	1212	649	671	717	988	914	776	839	603	
18	525	687	723	748	799	885	741	1212	649	671	717	988	914	776	839	603	
19	525	687	723	748	799	885	741	1212	649	671	717	988	581	776	839	603	
20	525	687	723	748	799	697	741	1212	649	493	717	617	581	776	839	603	
21	525	687	723	748	799	697	741	1212	649	493	717	617	581	776	839	603	
22	525	687	723	748	799	697	741	1212	649	493	717	988	581	776	839	603	
23	525	687	723	748	799	885	741	1212	649	671	717	988	581	776	839	603	
24	525	687	723	748	799	697	741	1212	649	493	717	988	914	776	839	603	
25	525	687	723	748	799	885	741	1212	649	671	717	988	914	776	839	603	
26	525	687	723	748	799	885	741	1212	862	493	717	988	581	776	839	603	
27	525	687	723	748	799	885	741	1212	649	493	717	617	581	776	839	603	
28	525	687	723	748	1112	697	741	1212	649	493	717	988	914	776	839	603	
29	525	687	723	748	799	885	741	1212	649	671	717	988	914	776	839	603	
30	525	687	723	748	799	885	741	1212	649	671	717	617	581	776	839	603	
31	525	687	723	748	1112	697	741	1212	649	493	717	988	914	776	839	603	
32	525	687	723	748	799	885	741	1212	862	671	717	988	581	776	839	603	
33	525	687	723	748	837	697	741	1212	649	493	717	988	914	776	839	603	
34	525	687	723	748	799	697	741	1212	649	493	717	617	581	776	839	603	
35	525	687	723	748	799	885	741	1212	649	493	717	988	581	776	839	603	
36																	

Figure A. 3: Precedence Matrix Used for Storing Shortest Path Information.

A.2 Description of Functions and Scripts

The tables given below have the description of the functions and scripts developed for calculation for travel time and travel cost.

Table A. 1: Description of Function for Calculation of Travel Time Based on Technology Type.

File Name:	**travel_time_any.m**
Purpose of Function:	Calculates the travel time for any one of the six technology types: A-90, A-110, A-125, A-150, HSR, Maglev.
Inputs:	distance, technology type ,maximum line speed allowed (0 for technology Default) and dwell time.
Outputs:	Travel time in seconds.
Calling functions:	None.

Table A. 2: Description of Function for Calculation of Travel Cost Based on Technology Type.

File Name:	**travelCost_any.m**
Purpose of Function:	Calculates the two-way travel cost for any one of the Six technology types: A-90, A-110, A-125, A-150, HSR, Maglev.
Inputs:	distances (accepts matrix or a single value) and technology type.
Outputs:	two-way travel cost in US dollars.
Calling functions:	None.

Table A. 3: Function for Preprocessing Connectivity/ Building Network using Route Information.

File Name:	**preprocess_connectivity.m**
Purpose of Function:	Builds the rail network from the given route files and station set.
Inputs:	Route file (struct file), stationcodes, technology type.
Outputs:	Traveltime, distances for connected nodes and connectivity matrix.
Calling functions:	travel_time_any.m

Table A. 4: Function for Performing Shortest Path Calculation.

File Name:	**floydTrain.m**
Purpose of Function:	Finds shortest path by travel time and corresponding distance for a given network.
Inputs:	Travel time and distances between directly connected origin-destination pairs.
Outputs:	Shortest travel time matrix with corresponding distances and Precedence matrix
Calling functions:	None.

Table A. 5: Function for Tracing Path Using the Precedence Matrix.

File Name:	**findPath.m**
Purpose of Function:	Traces the shortest path by using the Precedence matrix.
Inputs:	Origin and Destination node numbers/indices
Outputs:	Node indices on the shortest path.
Calling functions:	None.

Table A. 6: Script for Back-Calculation of Schedule Delay.

File Name:	**backCalculateSchedDelay.m**
Purpose of Script:	Calculates schedule delay for estimation of travel time without schedule delay
Inputs:	STATION_CODES10_25.mat, Sch1_CORRECT.mat,
Outputs:	scheudle_delay.mat (matrix)
Calling functions:	findPath.m

Table A. 7: Script for Running All Cases of Technology.

File Name:	**FullRun.m**
Purpose of Script:	Calculates total travel time and travel cost for all technology types.
Inputs:	STATION_CODES10_25.mat, Sch1_CORRECT.mat
Outputs:	Travel time, travel cost and distances for all technology types.
Calling functions:	preprocess_connectivity.m, floydTrain.m

Appendix B:

B.1 Source Code

1. **travel_time_any.m**

```
function [TT] = travel_time_any(dist,tech,max_speed,dwell)
% Technology Coefficients obtained by using speed time data from a report by US DOT
%Accepts distance, technology type and max_speed as inputs to give travel
%time Enter a_90, a_110, a_125, a_150 for Accelerail and HSR for high speed
%rail, Maglev for Maglev in the technology type argument. Enter a speed
%limit if needed ot enter 0 for default speed.
%dwell time is in minutes
% Called by preprocess_connectivity.m
stage_dist = dist;
technology = num2str(tech);
speed_limit = max_speed;
dwell_time = dwell;

switch lower(technology)
   case 'a_90'

      if speed_limit <= 0
         V_cruz = 90;              % default cruise velocity in mph
      elseif speed_limit > 0
         V_cruz = min(90,max_speed); %user entered cruise velocity in mph
      end

      decel = 0.75;              %deceleration in m/s^2
      accel = 0.75;              %acceleration in restricted cruise m/s^2
      alpha = 0.852653834;       %coefficient for technology
      beta = 0.008458217;        %coefficient for technology


      if stage_dist <= 0

         TTime = [];
         TTime = 0;

      elseif stage_dist > 0

         % find the time and distance for acceleration and deceleration for cruise
         % velocity

         uo = 0;

         u = V_cruz*0.44704 ; %convert speed to meters/second

         k = alpha/beta;

         a = (uo - k);
         b = (V_cruz - k);
```

```
        time_acc = (log(b) - log(a))/(-beta);  %seconds

        time_dec = (u/decel);                  %seconds

        dist_acc = 0.0001894 *(k*time_acc - ((k/beta)*(1-exp(-beta*time_acc)))); %+ ((uo/beta)*(1-exp(-
beta*time_acc)))   %miles

        dist_dec = 0.0006214*(u^2 / (2*decel));   % miles

        if (dist_acc + dist_dec) > stage_dist

           u1 =  sqrt(decel*1609.344*stage_dist);   %meters/sec

           t1 =  (u1/decel);              %seconds

           total_time  = 2*t1 ;           %seconds

           TTime = total_time  ;          %seconds

        else

           dist_cruz = (stage_dist - (dist_acc + dist_dec));  %miles

           time_cruz = (dist_cruz / V_cruz)*3600;              %seconds

           TTime = (time_acc + time_cruz + time_dec);         %seconds

        end


     end

     TT = TTime + (dwell_time*60);



     %disp('Method is linear')
  case 'a_110'

     if speed_limit <= 0
        V_cruz = 110;              % default cruise velocity in mph
     elseif speed_limit > 0
        V_cruz = min(110,max_speed); %user entered cruise velocity in mph
     end


     decel = 0.75;  %deceleration in m/s^2
     accel = 0.75;  %acceleration in restricted cruise m/s^2
     alpha = 1.254268603 ;       %coefficient for technology
     beta = 0.010151982  ;       %coefficient for technology


     if stage_dist <= 0

        TTime = [];
        TTime = 0;
```

```
    elseif stage_dist > 0

       % find the time and distance for acceleration and deceleration for cruise
       % velocity

       uo = 0;

       u = V_cruz*0.44704 ; %convert speed to meters/second

       k = alpha/beta;

       a = (uo - k);
       b = (V_cruz - k);

       time_acc = (log(b) - log(a))/(-beta);  %seconds

       time_dec = (u/decel);              %seconds

       dist_acc = 0.0001894 *(k*time_acc - ((k/beta)*(1-exp(-beta*time_acc)))); %+ ((uo/beta)*(1-exp(-
beta*time_acc)))   %miles

       dist_dec = 0.0006214*(u^2 / (2*decel));   % miles

       if (dist_acc + dist_dec) > stage_dist

          u1 =  sqrt(decel*1609.344*stage_dist);   %meters/sec

          t1 =  (u1/decel);             %seconds

          total_time  = 2*t1 ;          %seconds

          TTime = total_time  ;         %seconds

       else

          dist_cruz = (stage_dist - (dist_acc + dist_dec)) ; %miles

          time_cruz = (dist_cruz / V_cruz)*3600  ;             %seconds

          TTime = (time_acc + time_cruz + time_dec) ;        %seconds

       end

    end

    TT = TTime + (dwell_time*60); %seconds
    %disp('Method is cubic')

  case 'a_125'
    if speed_limit <= 0
       V_cruz = 125;              % default cruise velocity in mph
    elseif speed_limit > 0
       V_cruz = min(125,max_speed); %user entered cruise velocity in mph
    end
```

```
decel = 0.75;  %deceleration in m/s^2
accel = 0.75;  %acceleration in restricted cruise m/s^2
alpha = 1.880164198;        %coefficient for technology
beta = 0.013647527 ;        %coefficient for technology



if stage_dist <=0
   TTime = [];
   TTime = 0;
elseif stage_dist > 0

   % find the time and distance for acceleration and deceleration for cruise
   % velocity

   uo = 0;

   u = V_cruz*0.44704 ; %convert speed to meters/second

   k = alpha/beta;

   a = (uo - k);
   b = (V_cruz - k);

   time_acc = (log(b) - log(a))/(-beta);  %seconds

   time_dec = (u/decel);                  %seconds

   dist_acc = 0.0001894 *(k*time_acc - ((k/beta)*(1-exp(-beta*time_acc)))); %+ ((uo/beta)*(1-exp(-
beta*time_acc)))   %miles

   dist_dec = 0.0006214*(u^2 / (2*decel));   % miles

   if (dist_acc + dist_dec) > stage_dist

      u1 =  sqrt(decel*1609.344*stage_dist);   %meters/sec

      t1 =  (u1/decel) ;              %seconds

      total_time  = 2*t1 ;            %seconds

      TTime = total_time   ;          %seconds

   else

      dist_cruz = (stage_dist - (dist_acc + dist_dec)) ; %miles

      time_cruz = (dist_cruz / V_cruz)*3600    ;             %seconds

      TTime = (time_acc + time_cruz + time_dec)  ;        %seconds

   end
end

TT = TTime + (dwell_time*60); %seconds
```

```
    %disp('Method is nearest')

  case 'a_150'
    if speed_limit <= 0
       V_cruz = 150;              % default cruise velocity in mph
    elseif speed_limit > 0
       V_cruz = min(150,max_speed); %user entered cruise velocity in mph
    end

    decel = 0.75;  %deceleration in m/s^2
    accel = 0.75;  %acceleration in restricted cruise m/s^2
    alpha = 1.946582214;       %coefficient for technology
    beta =  0.011724359;       %coefficient for technology



    % find the time and distance for acceleration and deceleration for cruise
    % velocity

    uo = 0;

    u = V_cruz*0.44704 ; %convert speed to meters/second

    k = alpha/beta;

    a = (uo - k);
    b = (V_cruz - k);

    time_acc = (log(b) - log(a))/(-beta);  %seconds

    time_dec = (u/decel);                %seconds

    dist_acc = 0.0001894 *(k*time_acc - ((k/beta)*(1-exp(-beta*time_acc)))); %+ ((uo/beta)*(1-exp(-
beta*time_acc)))   %miles

    dist_dec = 0.0006214*(u^2 / (2*decel));   % miles

    if (dist_acc + dist_dec) > stage_dist

       u1 =  sqrt(decel*1609.344*stage_dist);   %meters/sec

       t1 =  (u1/decel);             %seconds

       total_time  = 2*t1  ;         %seconds

       TTime = total_time  ;         %seconds

    else

       dist_cruz = (stage_dist - (dist_acc + dist_dec)) ; %miles

       time_cruz = (dist_cruz / V_cruz)*3600 ;              %seconds

       TTime = (time_acc + time_cruz + time_dec) ;         %seconds
```

```
    end

    TT = TTime + (dwell_time*60); %seconds

  case 'hsr'
    if speed_limit <= 0
      V_cruz = 186;              % default cruise velocity in mph
    elseif speed_limit > 0
      V_cruz = min(186,max_speed); %user entered cruise velocity in mph
    end

    decel = 0.75;  %deceleration in m/s^2
    accel = 0.75;  %acceleration in restricted cruise m/s^2
    alpha = 1.401384247;       %coefficient for technology
    beta =  0.007008544 ;      %coefficient for technology


    if stage_dist <=0
      TTime = [];
      TTime = 0;

    elseif stage_dist > 0

      % find the time and distance for acceleration and deceleration for cruise
      % velocity

      uo = 0;

      u = V_cruz*0.44704 ; %convert speed to meters/second

      k = alpha/beta;

      a = (uo - k);
      b = (V_cruz - k);

      time_acc = (log(b) - log(a))/(-beta) ; %seconds

      time_dec = (u/decel) ;              %seconds

      dist_acc = 0.0001894 *(k*time_acc - ((k/beta)*(1-exp(-beta*time_acc)))); %+ ((uo/beta)*(1-exp(-
beta*time_acc)))   %miles

      dist_dec = 0.0006214*(u^2 / (2*decel));   % miles

      if (dist_acc + dist_dec) > stage_dist

        u1 =  sqrt(decel*1609.344*stage_dist) ;  %meters/sec

        t1 =  (u1/decel)  ;           %seconds

        total_time  = 2*t1  ;         %seconds

        TTime = total_time  ;         %seconds

      else
```

```
            dist_cruz = (stage_dist - (dist_acc + dist_dec));  %miles

            time_cruz = (dist_cruz / V_cruz)*3600 ;              %seconds

            TTime = (time_acc + time_cruz + time_dec) ;         %seconds

          end
        end


        TT = TTime + (dwell_time*60); %seconds

    case 'maglev'
        if speed_limit <= 0
            V_cruz = 300;               % default cruise velocity in mph
        elseif speed_limit > 0
            V_cruz = min(300,max_speed); %user entered cruise velocity in mph
        end

        decel = 0.75;  %deceleration in m/s^2
        accel = 0.75;  %acceleration in restricted cruise m/s^2
        alpha = 2.566397315;        %coefficient for technology
        beta =  0.006694159 ;       %coefficient for technology


        if stage_dist <= 0
            TTime = [];
            TTime =0;
        elseif stage_dist > 0

            % find the time and distance for acceleration and deceleration for cruise
            % velocity

            uo = 0;

            u = V_cruz*0.44704 ; %convert speed to meters/second

            k = alpha/beta;

            a = (uo - k);
            b = (V_cruz - k);

            time_acc = (log(b) - log(a))/(-beta);  %seconds

            time_dec = (u/decel);               %seconds

            dist_acc = 0.0001894 *(k*time_acc - ((k/beta)*(1-exp(-beta*time_acc)))); %+ ((uo/beta)*(1-exp(-
beta*time_acc)))   %miles

            dist_dec = 0.0006214*(u^2 / (2*decel));   % miles

            if (dist_acc + dist_dec) > stage_dist

                u1 =  sqrt(decel*1609.344*stage_dist) ;  %meters/sec

                t1 =  (u1/decel)   ;          %seconds
```

```
            total_time  = 2*t1 ;           %seconds

            TTime = total_time  ;          %seconds

         else

            dist_cruz = (stage_dist - (dist_acc + dist_dec)) ; %miles

            time_cruz = (dist_cruz / V_cruz)*3600   ;          %seconds

            TTime = (time_acc + time_cruz + time_dec) ;        %seconds

         end
      end

      TT = TTime + (dwell_time*60); %seconds

   otherwise
      disp('Unknown method.')
end
```

2. **travelCost_any.m**

```
function cost = travelCost_any(distance,tech)
%function gives two-way travel-cost for a given distance and technology
%Distance may be single value or matrix/array
%Called by FullRun.m

antar = distance;
lamb = length(antar);
technology = num2str(tech);
% Start a switch to perform required calculations based on user input.
switch lower(technology)

   case 'a_90'

      % Declare the calibrated constants
      a = -0.36479;
      b = 0.328033;
      c = 0.333028;


      % Perfrom calculations for the input distance matirx.
      for i = 1:lamb
         for j = 1:lamb
            if i~=j
               if antar(i,j) > 5
                  travel_cost(i,j) = antar(i,j)*(2*0.85*0.733*1/(a + b*(antar(i,j)^c)));
               else
                  travel_cost(i,j) = 5*(2*0.85*0.733*1/(a + b*(5^c)));
               end
            end
         end
```

```
        end

        cost = travel_cost;

    case 'a_110'

        % Declare the calibrated constants
        a = -0.36479;
        b = 0.328033;
        c = 0.333028;

%       travel_cost = 2*0.85*0.747*exp(a+(b./antar)+(c*log(antar))).*antar;

        % Perfrom calculations for the input distance matirx.
        for i = 1:lamb
            for j = 1:lamb
                if i~=j
                    if antar(i,j) > 5
                        travel_cost(i,j) = antar(i,j)*(2*0.85*0.747*1/(a + b*(antar(i,j)^c)));
                    else
                        travel_cost(i,j) = 5*(2*0.85*0.747*1/(a + b*(5^c)));
                    end
                end
            end
        end

        cost = travel_cost;

    case 'a_125'

        % Declare the calibrated constants
        a = -0.36479;
        b = 0.328033;
        c = 0.333028;

%       travel_cost = 2*0.85*0.822*exp(a+(b./antar)+(c*log(antar))).*antar;

        % Perfrom calculations for the input distance matirx.
        for i = 1:lamb
            for j = 1:lamb
                if i~=j
                    if antar(i,j) > 5
                        travel_cost(i,j) = antar(i,j)*(2*0.85*0.822*1/(a + b*(antar(i,j)^c)));
                    else
                        travel_cost(i,j) = 5*(2*0.85*0.822*1/(a + b*(5^c)));
                    end
                end
            end
        end
        cost = travel_cost;

    case 'a_150'

        % Declare the calibrated constants
        a = -0.36479;
        b = 0.328033;
```

```
    c = 0.333028;

%       travel_cost = 2*0.85*exp(a+(b./antar)+(c*log(antar))).*antar;

    % Perfrom calculations for the input distance matirx.
     for i = 1:lamb
        for j = 1:lamb
           if i~=j
              if antar(i,j) > 5
                 travel_cost(i,j) = antar(i,j)*(2*0.85*1/(a + b*(antar(i,j)^c)));
              else
                 travel_cost(i,j) = 5*(2*0.85*1/(a + b*(5^c)));
              end
           end
        end
     end

     cost = travel_cost;

  case 'hsr'

     % Declare the calibrated constants
     a = -0.36479;
     b = 0.328033;
     c = 0.333028;

%       travel_cost = 2*0.85*0.927*exp(a+(b./antar)+(c*log(antar))).*antar;

     % Perfrom calculations for the input distance matirx.
      for i = 1:lamb
        for j = 1:lamb
           if i~=j
              if antar(i,j) > 5
                 travel_cost(i,j) = antar(i,j)*(2*0.85*0.927*1/(a + b*(antar(i,j)^c)));
              else
                 travel_cost(i,j) = 5*(2*0.85*0.927*1/(a + b*(5^c)));
              end
           end
        end
      end

     cost = travel_cost;

  case 'maglev'

     % Declare the calibrated constants
     a = -0.36479;
     b = 0.328033;
     c = 0.333028;

%       travel_cost = 2*1.049*exp(a+(b./antar)+(c*log(antar))).*antar;

     % Perfrom calculations for the input distance matirx.
      for i = 1:lamb
        for j = 1:lamb
           if i~=j
```

```
            if antar(i,j) > 5
               travel_cost(i,j) = antar(i,j)*(2*1.049*1/(a + b*(antar(i,j)^c)));
            else
               travel_cost(i,j) = 5*(2*1.049*1/(a + b*(5^c)));
            end
         end
      end
   end

   cost = travel_cost;

otherwise
   disp('Unknown Technology.')
end
```

3. **preprocess_connectivity.m**

```
function [Traveltime distances connectivity] = preprocess_connectivity(sched,stationcodes,tech)

% Function to Process Travel time, distance and Connectivity of Rail
% network using Route structure, technology type and stationcodes

% Calls external function ---> %'travel_time_any',
%input for this function are-->
% distance,technology type,max line speed and dwelltime
% output is-->Travel time for the given distance.
% Called by FullRun.m
global Part_list
global schedule
%global stationcodes
tic
schedule = sched;
stn_codes = stationcodes;
 %dwell_time;
xrl = length(schedule); % find the length of the route structure
b = [];

% Generate a list of Line nodes of the type--> 'WAS12', 'NYP2' etc.
for i=1:length(schedule)
   schedule(i).stn_name_app = strcat(schedule(i).stn_name,num2str(i));
   temp_list = schedule(i).stn_name_app;
   b = [b;temp_list];
end

techno = tech;
% Build a complete list of stationcodes that includes line nodes and
% station/transfer nodes
Part_list = [stn_codes;b];
lrk = length(Part_list);
% Initialize distance and connectivity
conn =[];
dist = [];

dist = zeros(lrk);
```

```
conn = zeros(lrk);

% ------------------------------------------------------------------------%
% This part generates a distance matrix for all adjacent nodes using the
% route file specified by the user and the station list created in the
% previous step, additionally a dwell time and connectivity matrix is also
% generated.
% ------------------------------------------------------------------------%
for i=1:length(schedule)

   xx = length(schedule(i).stn_name);

   for j=1:(xx-1)

      n1 = schedule(i).stn_name_app(j); %Store adjacent stationID's from
      n2 = schedule(i).stn_name_app(j+1); %Route file in n1 and n2

      [c1,ia,ib] = intersect(n1,Part_list); %Find the index of these ID's
      [c2,ic,id] = intersect(n2,Part_list); % in the extended station ID list
      if isempty(c1)==0 && isempty(c2)==0   % Check if both the ID's are found

         % using the mapped index values, calculate the distance between
         % the two adjacent station ID's
         dist(ib,id) = abs(schedule(i).distance(j)-schedule(i).distance(j+1));
         conn(ib,id)  = 1 ; %Give a connectivity value of 1 to represent adjacency
         dwell(ib,id) =  schedule(i).dwell_time;
         % same as the above step with flipped ID's, since this is an
         % undirected graph.
         dist(id,ib) = abs(schedule(i).distance(j)-schedule(i).distance(j+1));
         conn(id,ib)= 1 ;
         dwell(id,ib) =  schedule(i).dwell_time;
      end

   end
end
%------------------------------------------------------------------------%

%This part is meant for loading the travel time values on the previously
%obtained distance matrix based on the technology type, maximum line speed
%and dwell time, external function called in this step is--->
%'travel_time_any' input for this function are-->
% distance,technology type,max line speed and dwelltime, output is-->Travel
% time for the given distance.
%------------------------------------------------------------------------%
for i=1:length(Part_list)
   for j=1:length(Part_list)
      if dist(i,j) > 0 && i~=j
         travel_time_test(i,j) = travel_time_any(dist(i,j),techno,0,dwell(i,j)); % travel time is in seconds
      elseif dist(i,j) <= 0 && i~=j
         travel_time_test(i,j)= 9999999;  % High Travel time penalty for non-adjacent nodes
      end
   end
end

%------------------------------------------------------------------------%
%This part is used to insert the station/transfer nodes in the network,
```

```
%this step connects all the subgraphs created in the above step to form a
%single graph.

for i=1:length(schedule)
   xx = length(schedule(i).stn_name);
   for j=1:xx
      n1 = schedule(i).stn_name_app(j);  % Store the station ID with route number in n1(Line node)
      n2 = schedule(i).stn_name(j);      %Store the station ID without route number in n2(station/transfer
node)

      [c1,ia,ib] = intersect(n1,Part_list); %Find the index of these ID's
      [c2,ic,id] = intersect(n2,Part_list); % in the extended station ID list
      if isempty(c1)==0 && isempty(c2)==0   % Check if both the ID's are found

         travel_time_test(ib,id) = 0;      % Give a zero travel time value for entry into the station node
         conn(ib,id)  = 1 ;                % Specify connectivity

         travel_time_test(id,ib) = (3600*4)/(schedule(i).frequency); % Give a travel time value of schedule
delay for exiting station node
         conn(id,ib)= 1 ;                  % Specify connectivity

      end
   end
end
%-------------------------------------------------------------------------%

%Delcare output values
distances = dist;
connectivity = conn;
Traveltime =  travel_time_test;

Toc
```

4. **floydTrain.m**

```
function [travel_time distmat_cost Preced] = floydTrain(time_mat, distod)
% This function accepts a matrix containing travel time and distances
% between O-D pairs and returns a matrices containing shortest travel times
% and corresponding distance with the shortest path stored in form of a
% Precedence matrix.
% Called by FullRun.m
global P

travel_time_test  = time_mat;
od_dist1 = distod;

n = length(travel_time_test);

for i=1:n
   for j=1:n
      if od_dist1(i,j) > 4000
         od_dist1(i,j) = 0;
      end
   end
end
```

```
% n = length(travel_time_test);

P = ones(n,n);
t = 1:n;
for i=1:n
   P(i,:) = t(i).*P(i,:);
end

A = travel_time_test;

'Running Floyd for Travel time'
%Code for Floyd's algorithm, this uses a matrix 'A' with travel time as
%edge weights
tic
for k=1:n
   for i=1:n
      for j=1:n
         if A(i,k)+ A(k,j) < A(i,j)  % Check if insertion of node k improves the existing value
            A(i,j) = A(i,k) + A(k,j); % If there is an improvement, the distance is updated
            od_dist1(i,j) = od_dist1(i,k) + od_dist1(k,j); % the corresponding distance is also updated
            P(i,j)= P(k,j);        % k is included in the shortest path precedence matrix
         end
      end
   end

end
toc
'Done'

Preced = P;
travel_time = A;

distmat_cost = od_dist1;
```

5. **findPath.m**

```
function [route] = findPath(org, dest)
% Function to obtain path stored in the Precedence Matrix
% Called by backCalculateSchedDelay.m

global stationcodes  % Declare global variables since these have been created
global P             % in the workspace as global P--> Precedence matrix
origin= org;
destination= dest;

link = destination;      % Define the last node in the path
currentj = destination;  % Set the current node value to destination node
                    % last node in the path
while P(origin,currentj) ~= origin % Access the nodes till the current node becomes
                           % the same as the origin node
   prenode = P(origin,currentj);

   link = [link;prenode]; %Add the preceding node to the list of nodes on the shortest path
   currentj = prenode;    %assign preceding node as current node
```

```
end
link = [link;origin];      % add the origin node to the list of nodes on the shortest path
link = flipud(link);  %Link gives the shortest path but in the reverse order, flipud flips the
route = link;         %the list of nodes to return the actual shortest path.
```

6. **backCalculateSchedDelay.m**

```
%Script for backcalculating the schedule delay
% Calls findPath.m
global P
tic

load STATION_CODES10_25.mat  %loads up the file for stationcodes, variable name--> stationcodes
load Sch1_CORRECT.mat        %loads the struct file for route specification, variable name--> Sch1

% Create a field in the route structure that has a list of all the line nodes in the network,
%this includes line nodes eg. (BWI12)

b = [];
for i=1:length(Sch1)
   Sch1(i).stn_name_app = strcat(Sch1(i).stn_name,num2str(i));  % concatenate the route number to the
   temp_list = Sch1(i).stn_name_app;                           % node number to generate line nodes
   b = [b;temp_list];                                  % a line node for a station WAS  contained in Route
end                                                    % number 4 will be--> WAS4

% Create a list of all the nodes in the network, this includes line nodes
% eg. (BWI12) and Transfer nodes eg. (BWI)

Part_list = [stationcodes;b];
scheudle_delay =[];


for i = 1:length(stationcodes)
   for j = 1:length(stationcodes)
      if i~=j
         pathOrgDest = findPath(i,j);                  % traces the path from an origin to a destination
         firstStn = Part_list(pathOrgDest(2));           % map the node numbers to the actual stationcodes
         firstStnName = char(firstStn);                % held in Part_list
         lets = length(firstStnName);
         if length(firstStnName) == 4                  % this step checks if the length of the line node
            schedNo = str2num(firstStnName(4));              % if the length is found to be 4 it means the node
         elseif length(firstStnName) > 4               % is of the type --> 'WAS5','NYP4' etc., thus the
route number
            schedNo = str2num(firstStnName((lets-1):lets));     % for the  node is the last character of the
string array,
                                                       % otherwise the node is of the type 'NYP30', 'NYP12' etc.,
         end                                             % thus the route number is the last two characters in the string
         scheudle_delay(i,j) = (4/Sch1(schedNo).frequency);      % works for route numbers till 99

      end
   end
   toc
end
```

7. **FullRun.m**

```
%Script for running all Total travel time and travel cost cases in one go
% Calls preprocess_connectivity.m, floydTrain
load STATION_CODES10_25.mat
load Sch1_CORRECT.mat

CoreNodes = length(stationcodes);

[Traveltime distances connectivity] = preprocess_connectivity(Sch1,stationcodes,'a_90'); % Obtain Travel
time, distance and Connectivity
                                                        % Matrices for Network analysis
[TT_A90Full Dist_A90Full P] = floydTrain(Traveltime, distances);                % Call Floyd's
Algorithm for network analysis

TT_A90 = TT_A90Full(1:CoreNodes,1:CoreNodes)/3600;                              % Create a Matirx for
the core nodes
Dist_A90 = Dist_A90Full(1:CoreNodes,1:CoreNodes);
TC_A90 = travelCost_any(Dist_A90,'a_90');

save('C:\MATLAB7\work\ResultsNov_1\A_90\TT_A90','TT_A90')                       % Save the data
save('C:\MATLAB7\work\ResultsNov_1\A_90\TC_A90','TC_A90')
save('C:\MATLAB7\work\ResultsNov_1\A_90\TT_A90Full','TT_A90Full')
save('C:\MATLAB7\work\ResultsNov_1\A_90\Dist_A90','Dist_A90')
save('C:\MATLAB7\work\ResultsNov_1\A_90\Dist_A90Full','Dist_A90Full')
save('C:\MATLAB7\work\ResultsNov_1\A_90\P','P')

clear all

'DONE-A90'


load STATION_CODES10_25.mat
load Sch1_CORRECT.mat

CoreNodes = length(stationcodes);
[Traveltime distances connectivity] = preprocess_connectivity(Sch1,stationcodes,'a_110');
[TT_A110Full Dist_A110Full P] = floydTrain(Traveltime, distances);

TT_A110 = TT_A110Full(1:CoreNodes,1:CoreNodes)/3600;
Dist_A110 = Dist_A110Full(1:CoreNodes,1:CoreNodes);
TC_A110 = travelCost_any(Dist_A110,'a_110');

save('C:\MATLAB7\work\ResultsNov_1\A_110\TT_A110','TT_A110')
save('C:\MATLAB7\work\ResultsNov_1\A_110\TC_A110','TC_A110')
save('C:\MATLAB7\work\ResultsNov_1\A_110\TT_A110Full','TT_A110Full')
save('C:\MATLAB7\work\ResultsNov_1\A_110\Dist_A110','Dist_A110')
save('C:\MATLAB7\work\ResultsNov_1\A_110\Dist_A110Full','Dist_A110Full')
save('C:\MATLAB7\work\ResultsNov_1\A_110\P','P')

clear all
```

```
'DONE-A110'


load STATION_CODES10_25.mat
load Sch1_CORRECT.mat

CoreNodes = length(stationcodes);
[Traveltime distances connectivity] = preprocess_connectivity(Sch1,stationcodes,'a_125');
[TT_A125Full Dist_A125Full P] = floydTrain(Traveltime, distances);

TT_A125 = TT_A125Full(1:CoreNodes,1:CoreNodes)/3600;
Dist_A125 = Dist_A125Full(1:CoreNodes,1:CoreNodes);
TC_A125 = travelCost_any(Dist_A125,'a_125');

save('C:\MATLAB7\work\ResultsNov_1\A_125\TT_A125','TT_A125')
save('C:\MATLAB7\work\ResultsNov_1\A_125\TC_A125','TC_A125')
save('C:\MATLAB7\work\ResultsNov_1\A_125\TT_A125Full','TT_A125Full')
save('C:\MATLAB7\work\ResultsNov_1\A_125\Dist_A125','Dist_A125')
save('C:\MATLAB7\work\ResultsNov_1\A_125\Dist_A125Full','Dist_A125Full')
save('C:\MATLAB7\work\ResultsNov_1\A_125\P','P')

clear all


'DONE-A125'


load STATION_CODES10_25.mat
load Sch1_CORRECT.mat

CoreNodes = length(stationcodes);
[Traveltime distances connectivity] = preprocess_connectivity(Sch1,stationcodes,'a_150');
[TT_A150Full Dist_A150Full P] = floydTrain(Traveltime, distances);

TT_A150 = TT_A150Full(1:CoreNodes,1:CoreNodes)/3600;
Dist_A150 = Dist_A150Full(1:CoreNodes,1:CoreNodes);
TC_A150 = travelCost_any(Dist_A150,'a_150');

save('C:\MATLAB7\work\ResultsNov_1\A_150\TT_A150','TT_A150')
save('C:\MATLAB7\work\ResultsNov_1\A_150\TC_A150','TC_A150')
save('C:\MATLAB7\work\ResultsNov_1\A_150\TT_A150Full','TT_A150Full')
save('C:\MATLAB7\work\ResultsNov_1\A_150\Dist_A150','Dist_A150')
save('C:\MATLAB7\work\ResultsNov_1\A_150\Dist_A150Full','Dist_A150Full')
save('C:\MATLAB7\work\ResultsNov_1\A_150\P','P')

clear all

'DONE-A150'


load STATION_CODES10_25.mat
load Sch1_CORRECT.mat

CoreNodes = length(stationcodes);
[Traveltime distances connectivity] = preprocess_connectivity(Sch1,stationcodes,'hsr');
```

```
[TT_HSRFull Dist_HSRFull P] = floydTrain(Traveltime, distances);

TT_HSR = TT_HSRFull(1:CoreNodes,1:CoreNodes)/3600;
Dist_HSR = Dist_HSRFull(1:CoreNodes,1:CoreNodes);
TC_HSR = travelCost_any(Dist_HSR,'hsr');


save('C:\MATLAB7\work\ResultsNov_1\HSR\TT_HSR','TT_HSR')
save('C:\MATLAB7\work\ResultsNov_1\HSR\TC_HSR','TC_HSR')
save('C:\MATLAB7\work\ResultsNov_1\HSR\TT_HSRFull','TT_HSRFull')
save('C:\MATLAB7\work\ResultsNov_1\HSR\Dist_HSR','Dist_HSR')
save('C:\MATLAB7\work\ResultsNov_1\HSR\Dist_HSRFull','Dist_HSRFull')
save('C:\MATLAB7\work\ResultsNov_1\HSR\P','P')

clear all

'DONE-HSR'



load STATION_CODES10_25.mat
load Sch1_CORRECT.mat

CoreNodes = length(stationcodes);
[Traveltime distances connectivity] = preprocess_connectivity(Sch1,stationcodes,'maglev');
[TT_maglevFull Dist_maglevFull P] = floydTrain(Traveltime, distances);

TT_MAG = TT_maglevFull(1:CoreNodes,1:CoreNodes)/3600;
Dist_MAG = Dist_maglevFull(1:CoreNodes,1:CoreNodes);
TC_MAG = travelCost_any(Dist_MAG,'maglev');

save('C:\MATLAB7\work\ResultsNov_1\Maglev\TT_MAG','TT_MAG')
save('C:\MATLAB7\work\ResultsNov_1\Maglev\TC_MAG','TC_MAG')
save('C:\MATLAB7\work\ResultsNov_1\Maglev\TT_maglevFull','TT_maglevFull')
save('C:\MATLAB7\work\ResultsNov_1\Maglev\Dist_MAG','Dist_MAG')
save('C:\MATLAB7\work\ResultsNov_1\Maglev\Dist_maglevFull','Dist_maglevFull')
save('C:\MATLAB7\work\ResultsNov_1\Maglev\P','P')

clear all


'DONE-MAGLEV'
```

Appendix C:

Validation plots of travel time, station to station distances and speeds for the remaining technology types are presented section.

C.1 Performance Function Validation

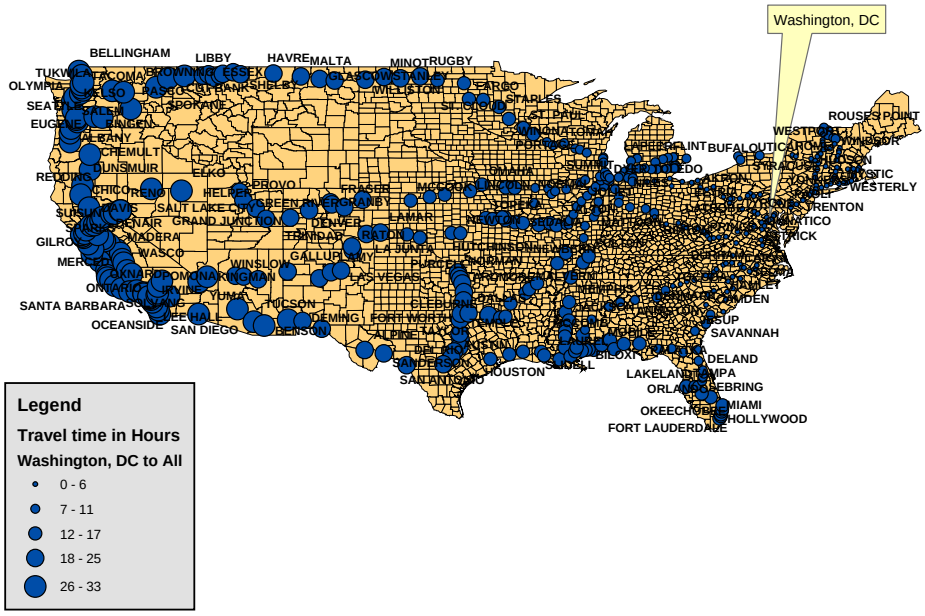


Figure C 1: Acela-110, DC to All.

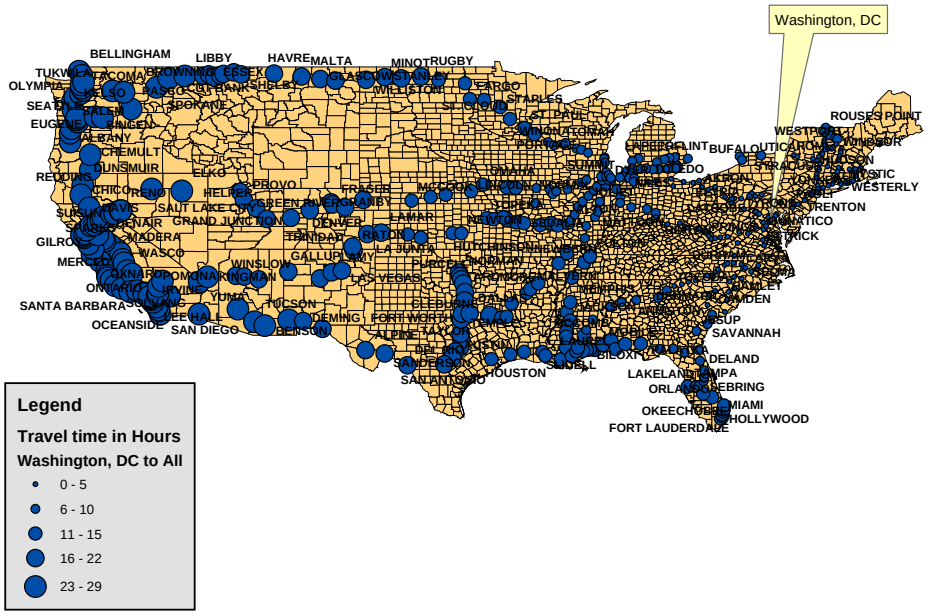


Figure C 2: Acela-125, DC to All.

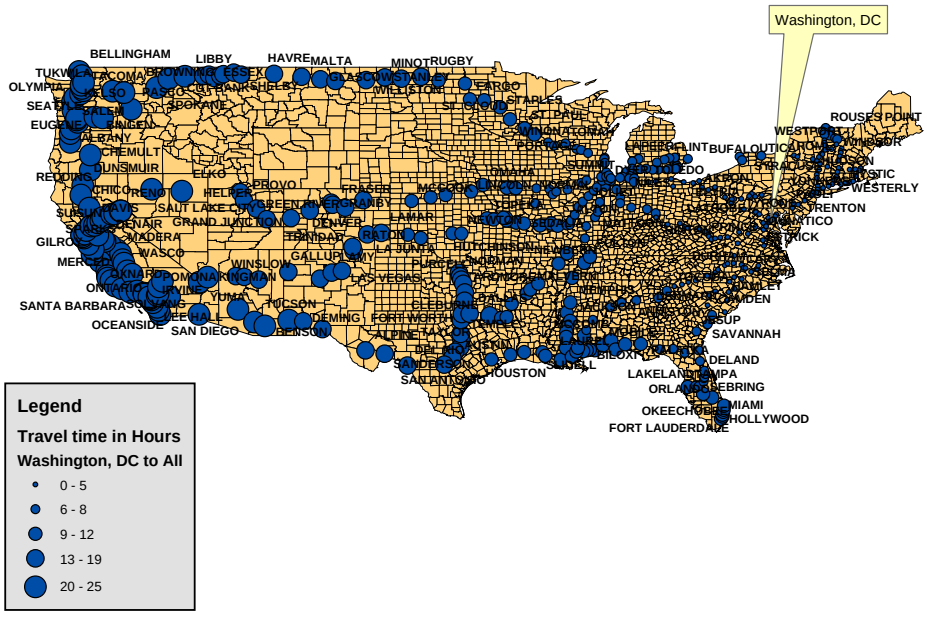


Figure C 3: Acela-150, DC to All.

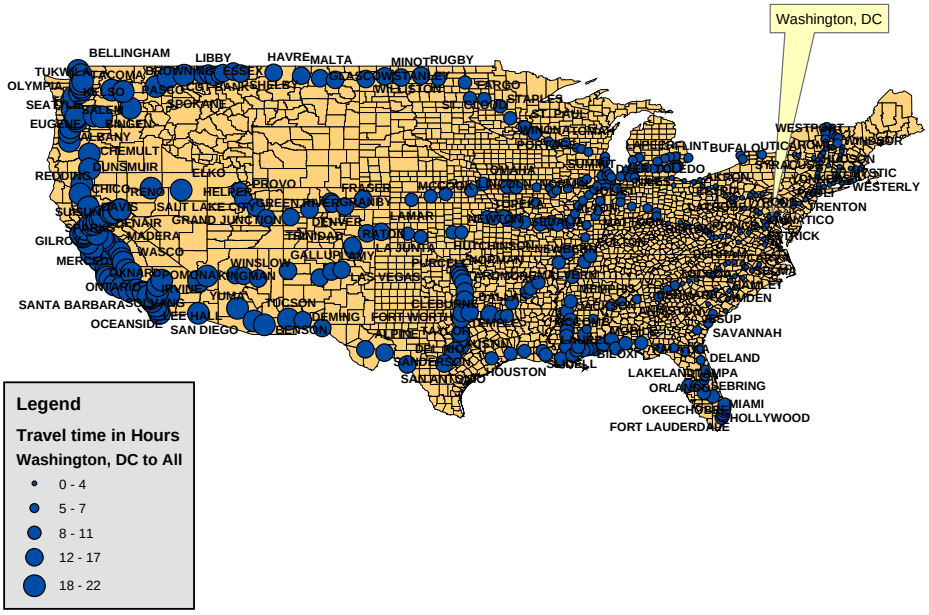


Figure C 4: High Speed Rail, DC to All.

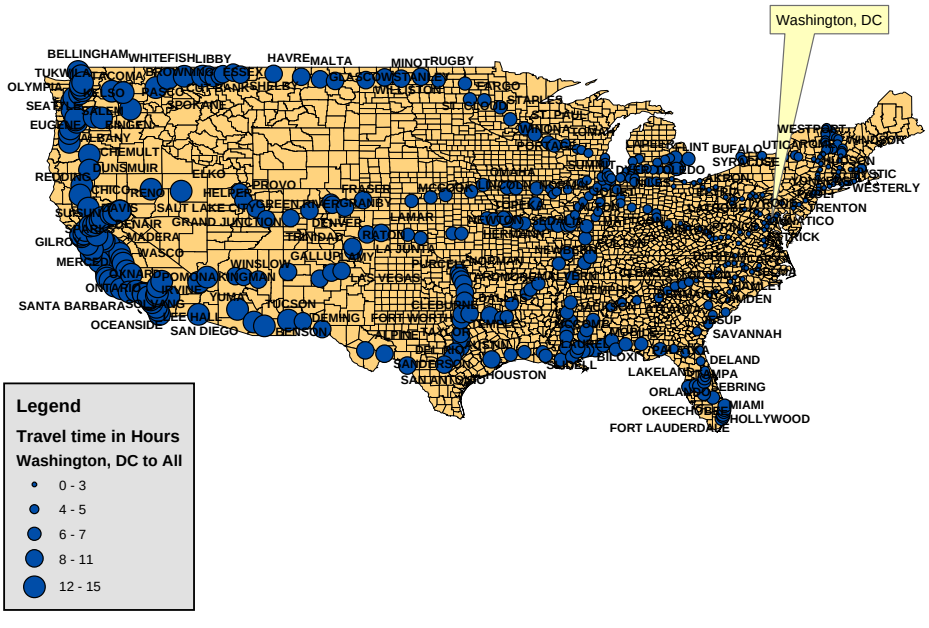


Figure C 5: Maglev, DC to All.

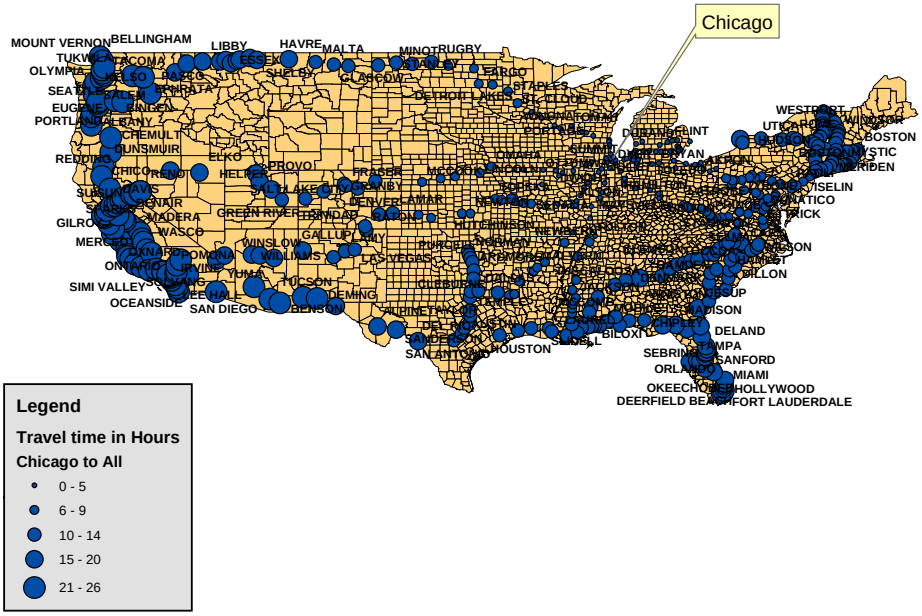


Figure C 6: Acela-110, Chicago to All.

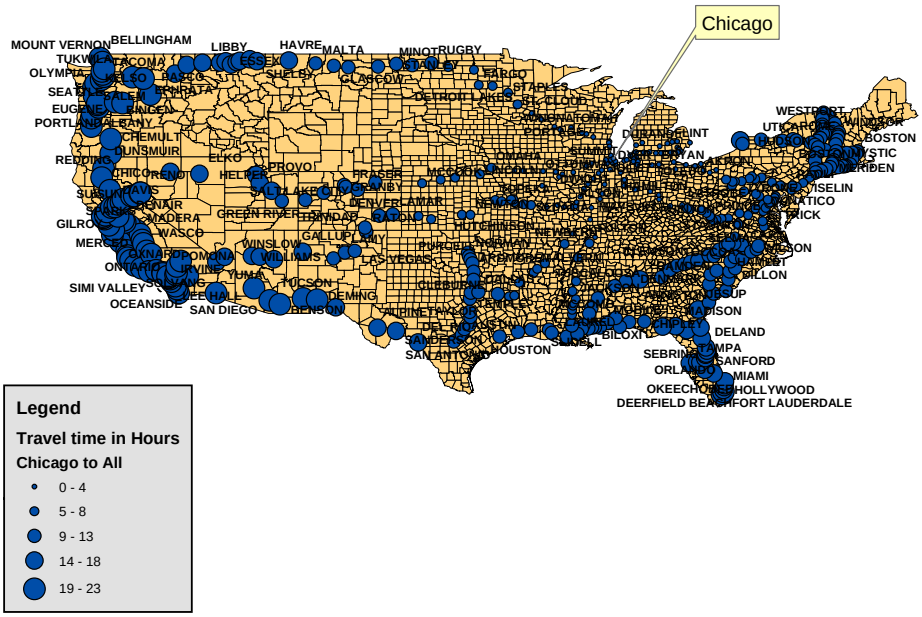


Figure C 7: Acela-125, Chicago to All.

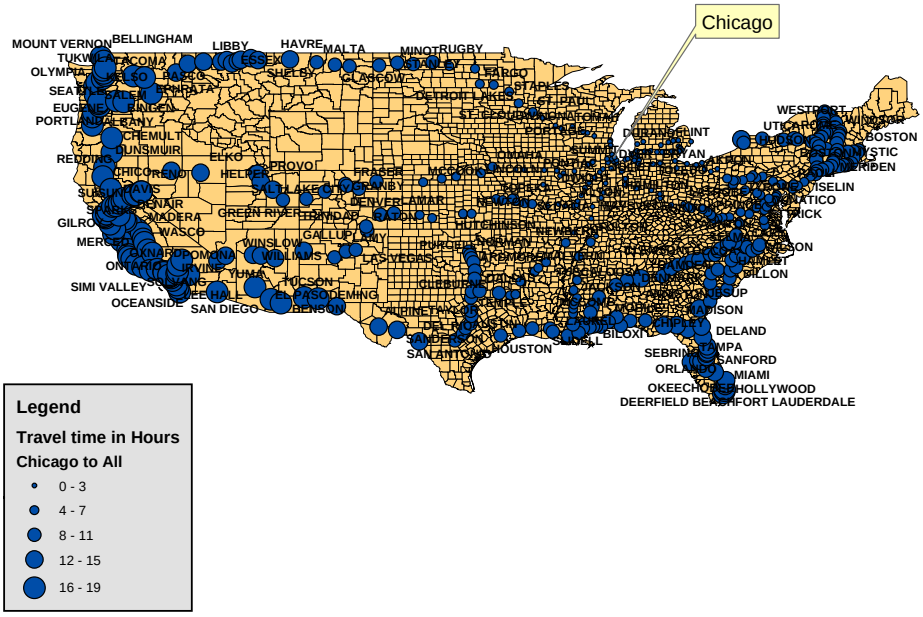


Figure C 8: Acela-150, Chicago to All.

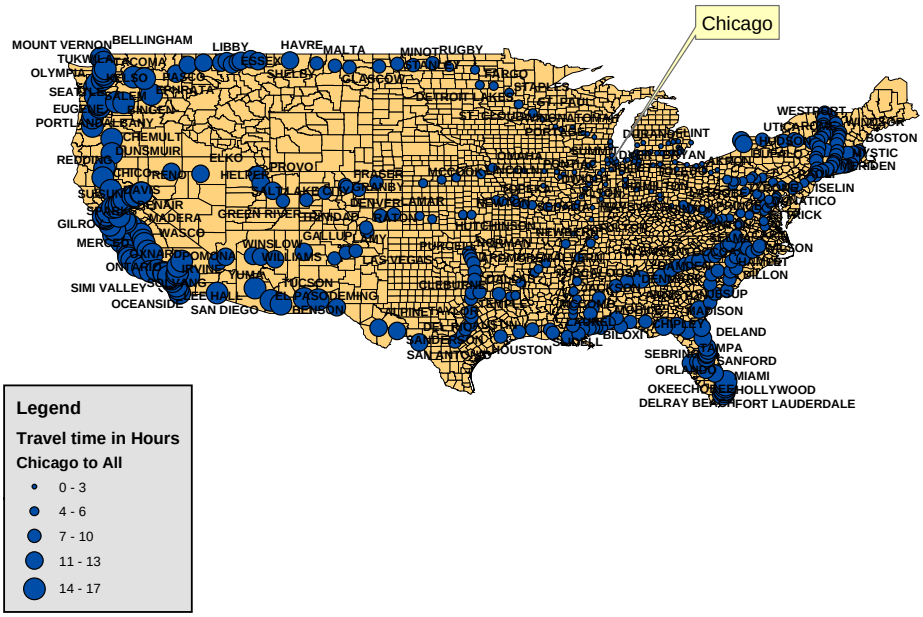


Figure C 9: High Speed Rail, Chicago to All.

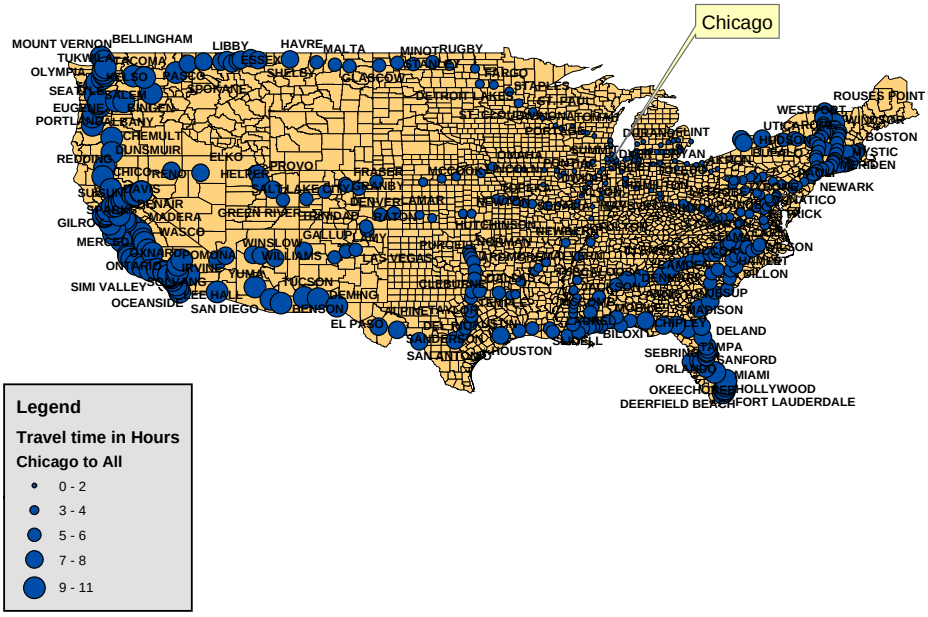


Figure C 10: Maglev, Chicago to All.

C.2 Travel Time Validation Plots

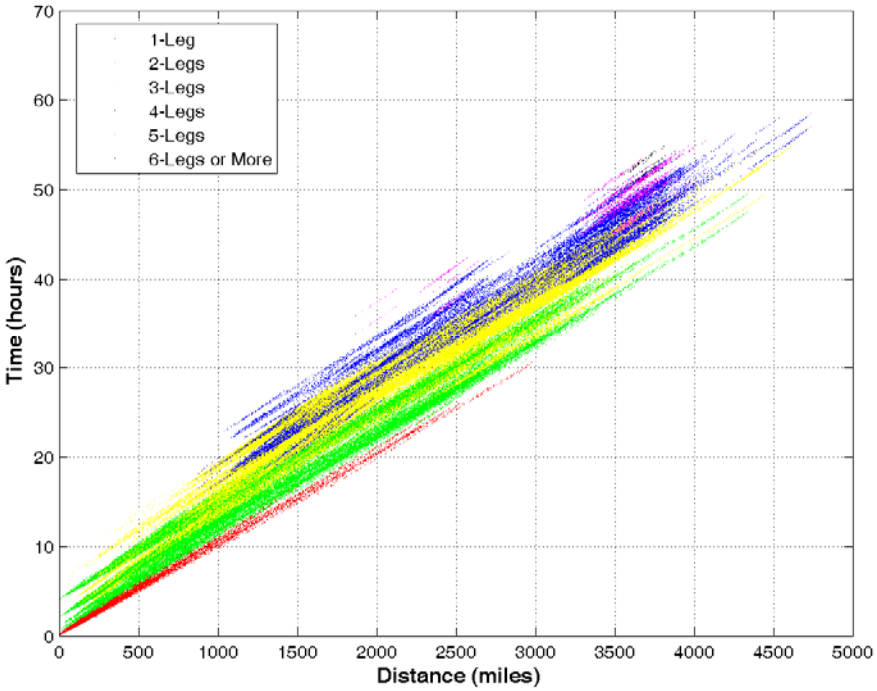


Figure C 11: Time v/s Distance Plot for Acela 110.

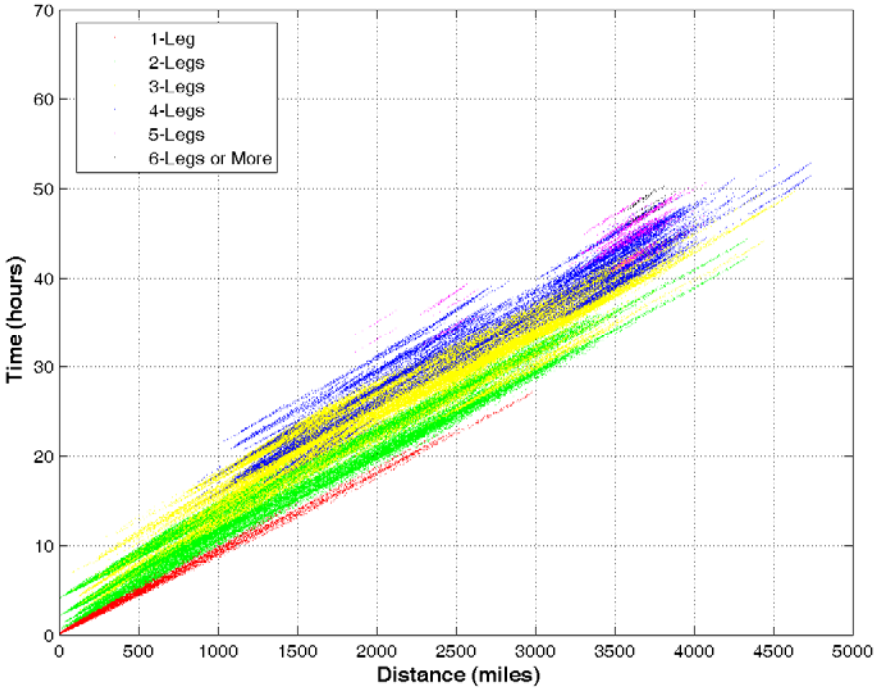


Figure C 12: Time v/s Distance Plot for Acela 125.

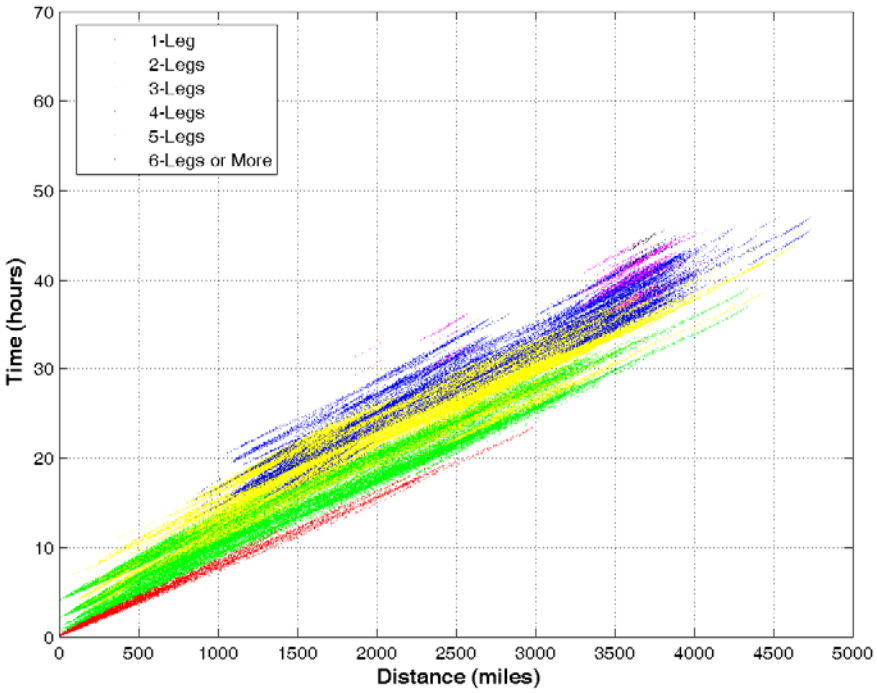


Figure C 13: Time v/s Distance Plot for Acela 150.

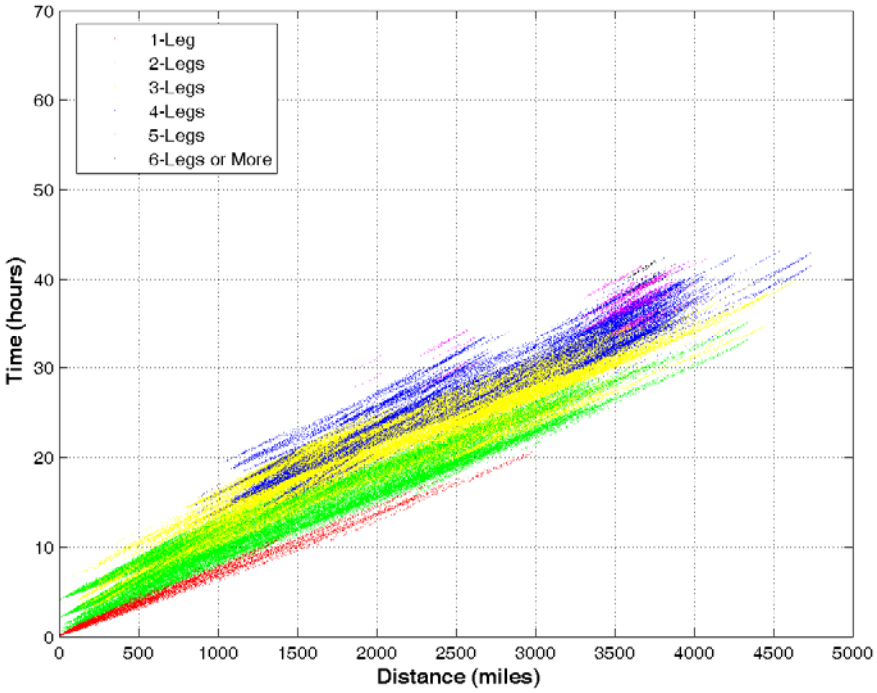


Figure C 14: Time v/s Distance Plot for High Speed Rail.

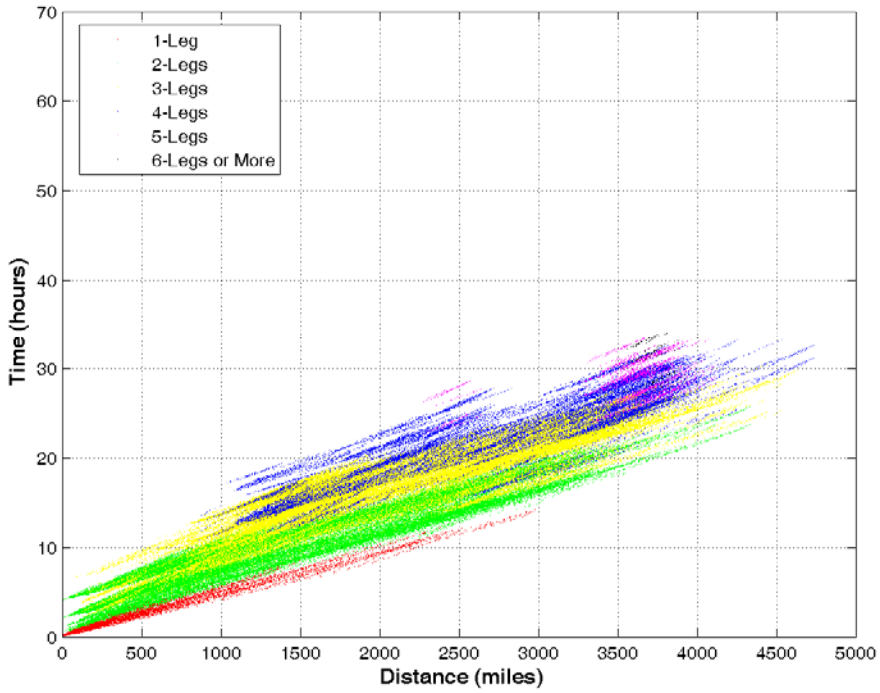


Figure C 15: Time v/s Distance Plot for Maglev.

C.3 Speed Profile Plots

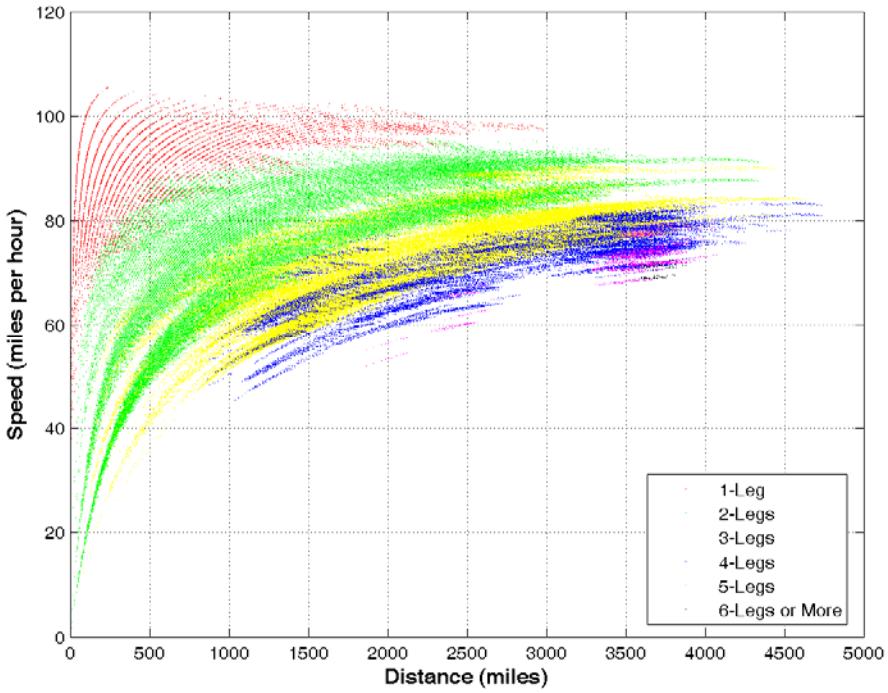


Figure C 16: Speed v/s Distance Curves for Acela 110.

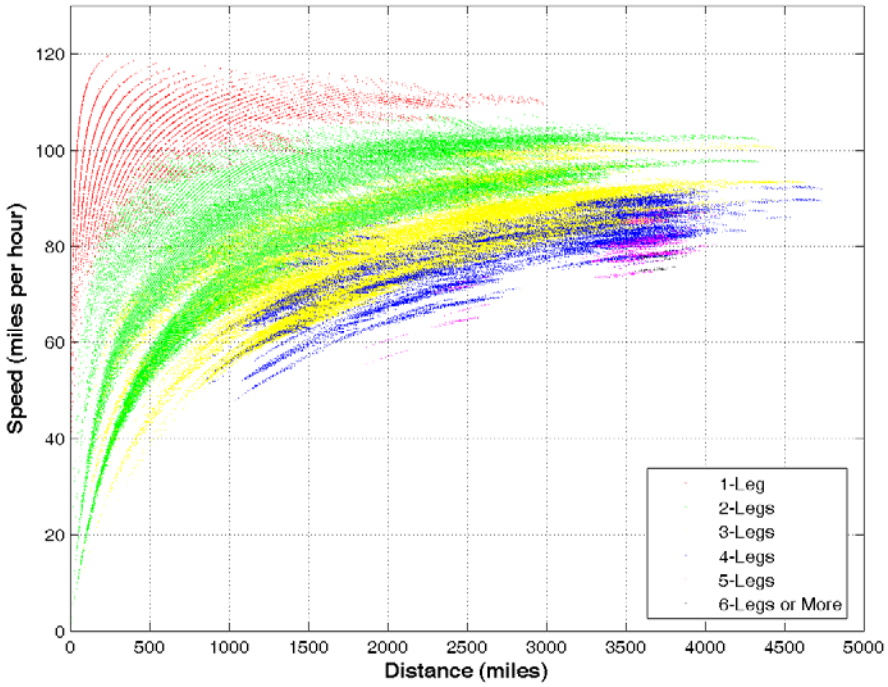


Figure C 17: Speed v/s Distance Curves for Acela 125.

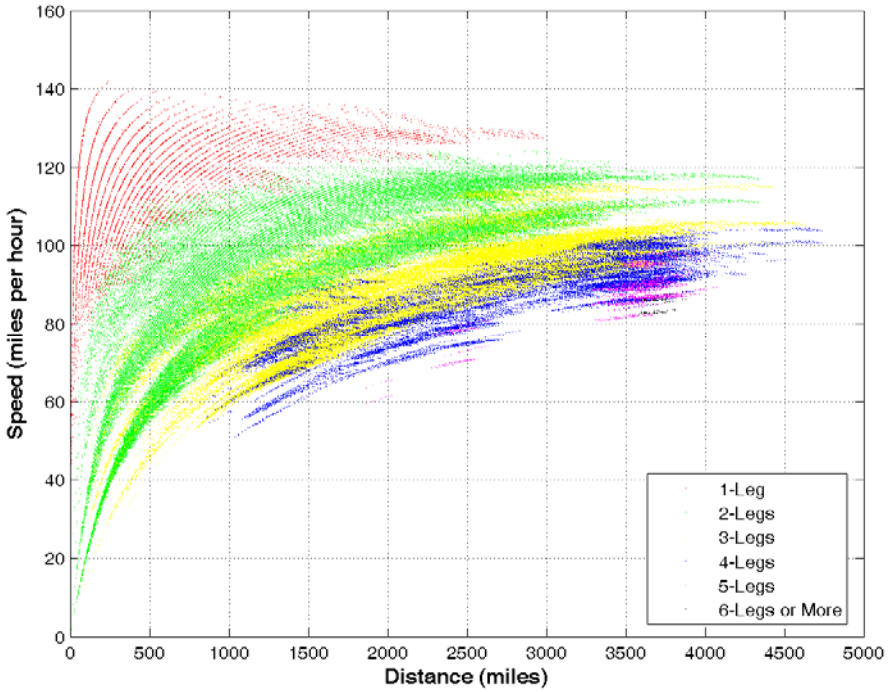


Figure C 18: Speed v/s Distance Curves for Acela 150.

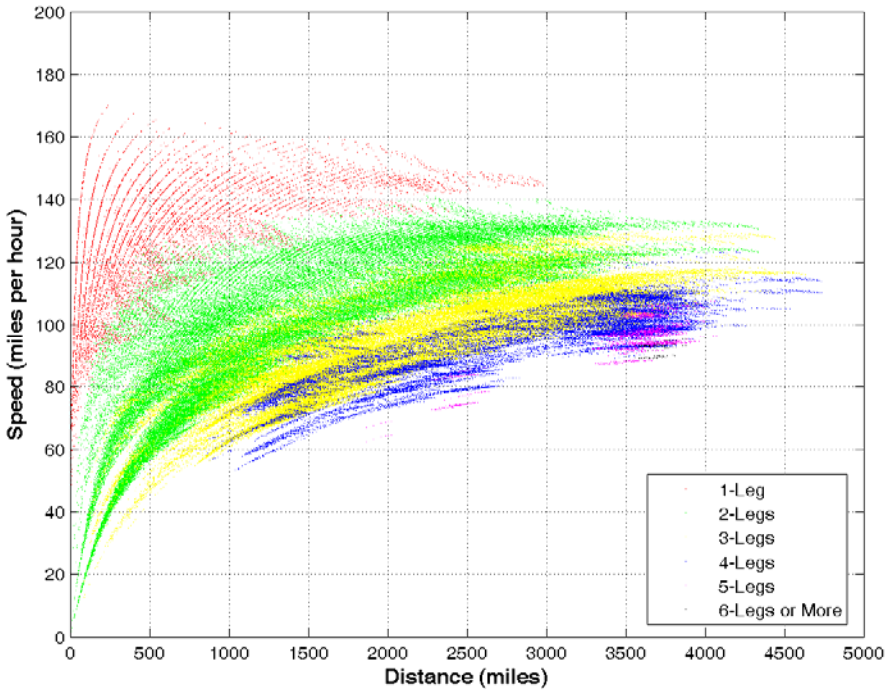


Figure C 19: Speed v/s Distance Curve for High Speed Rail.

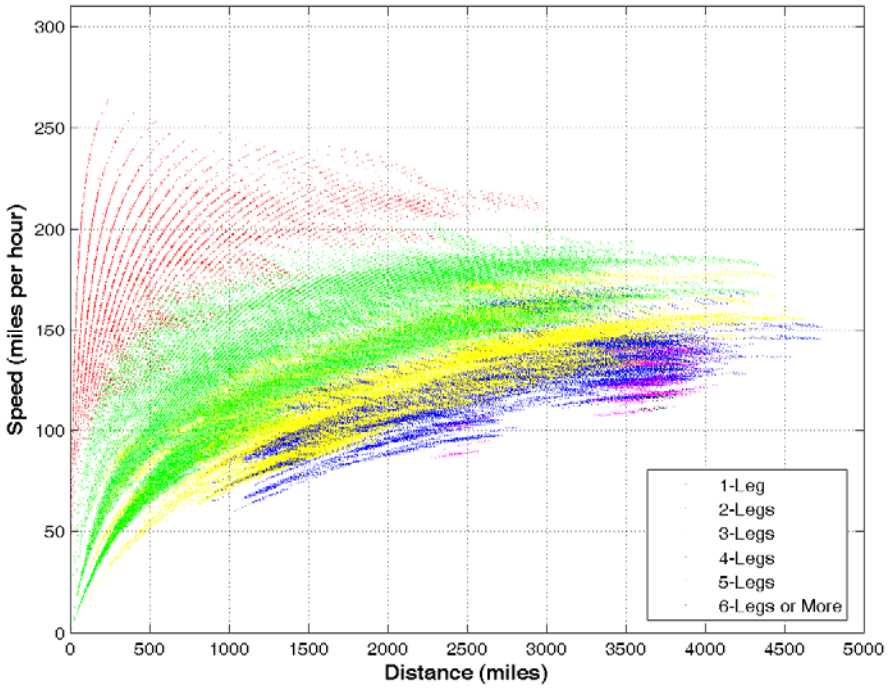


Figure C 20: Speed v/s Distance Curve for Maglev.

C.4 Distribution of Speeds

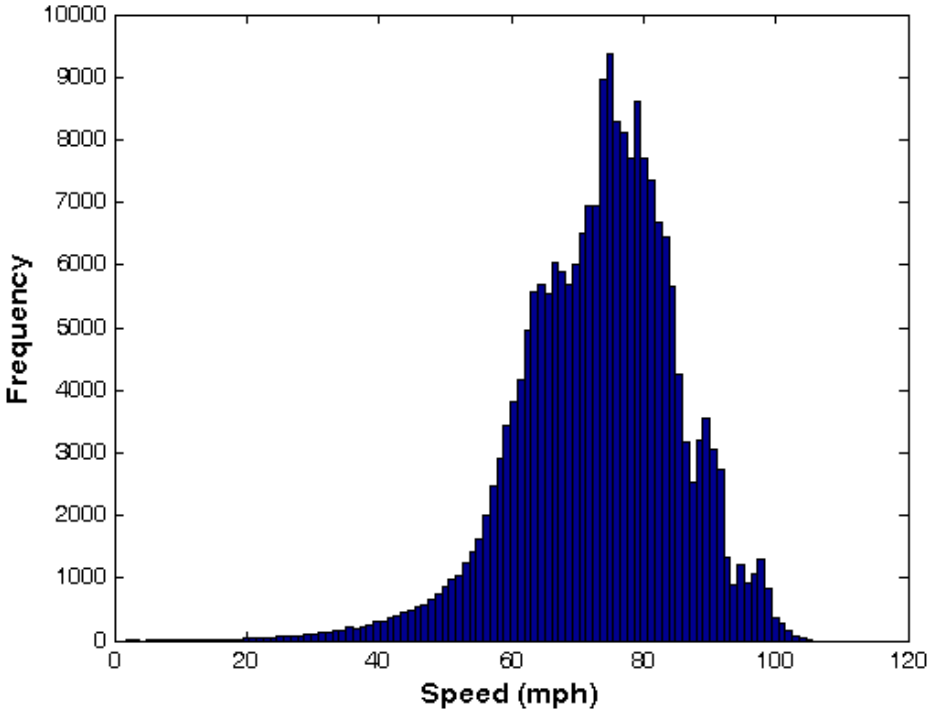


Figure C 21: Distribution of Speeds for Acela 110.

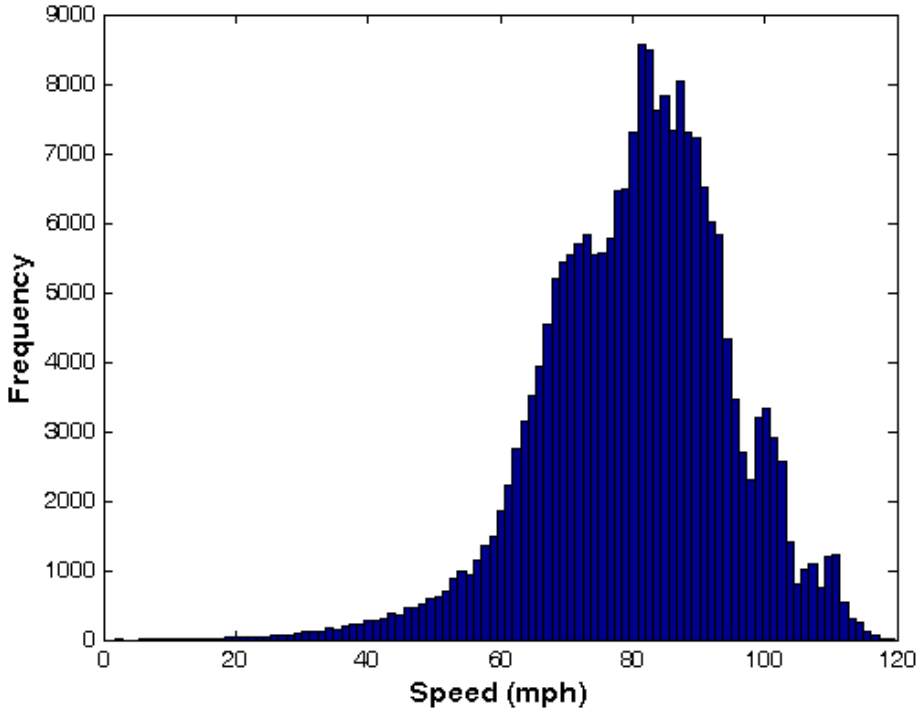


Figure C 22: Distribution of Speed for Acela 125.

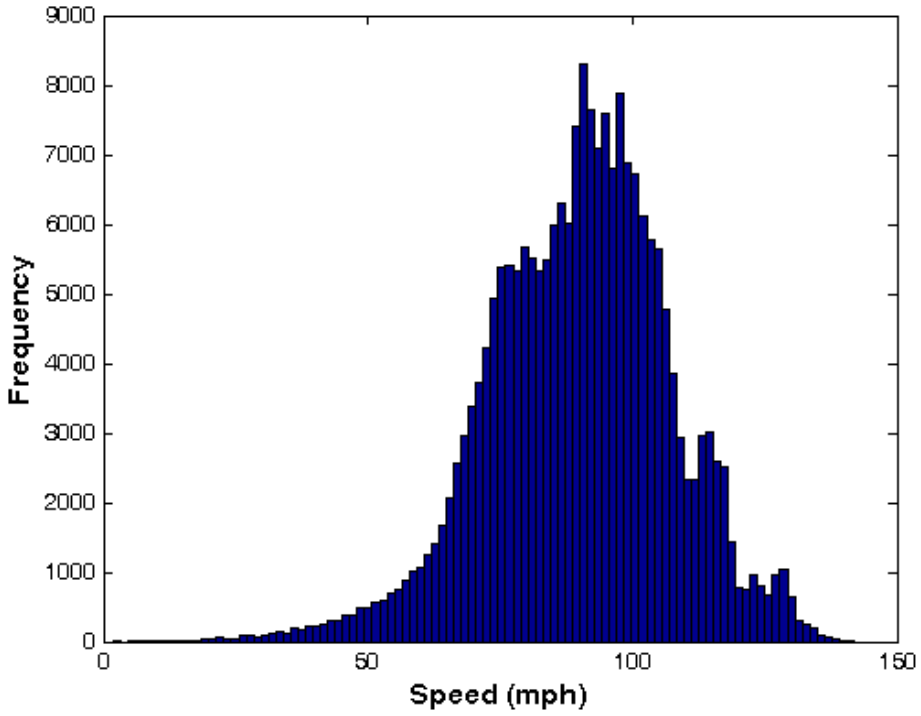


Figure C 23: Distribution of Speeds for Acela-150.

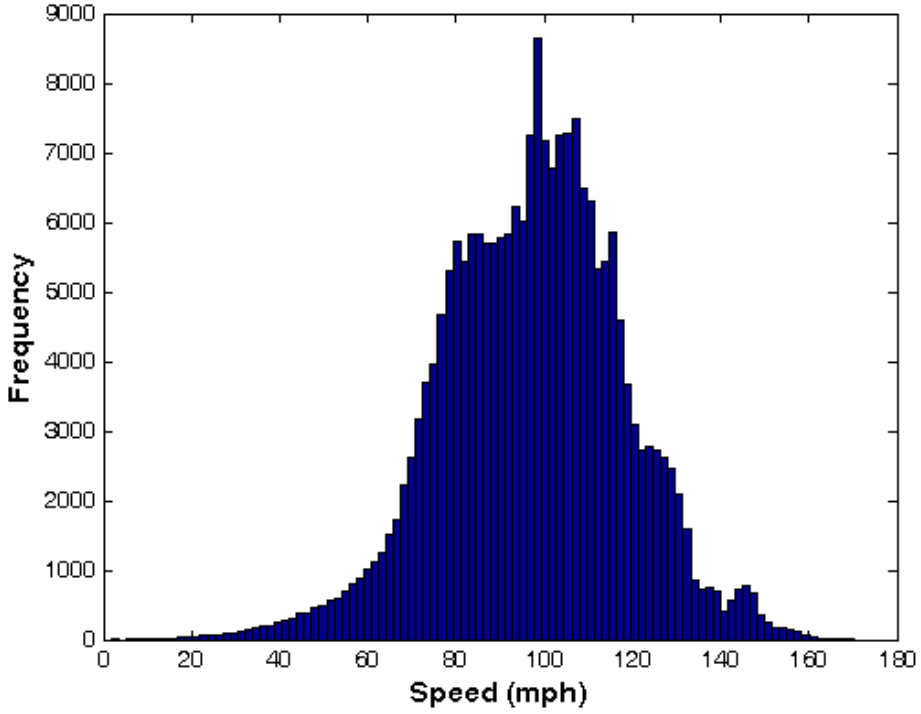


Figure C 24: Distribution of Speeds for High Speed Rail.

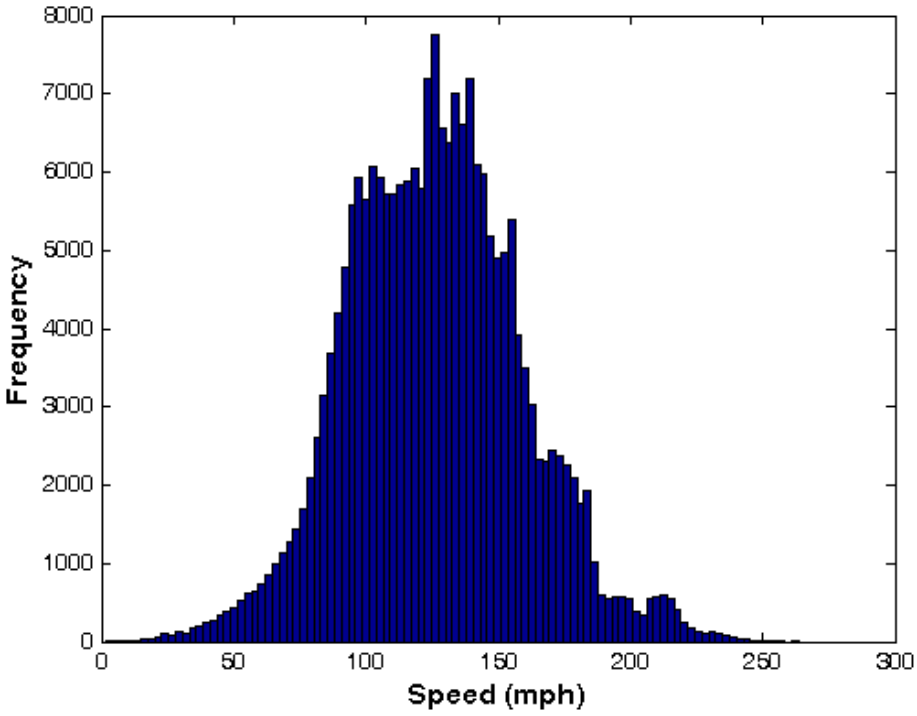


Figure C 25: Distribution of Speeds for Maglev.

C.5 Distribution of Detour Factors

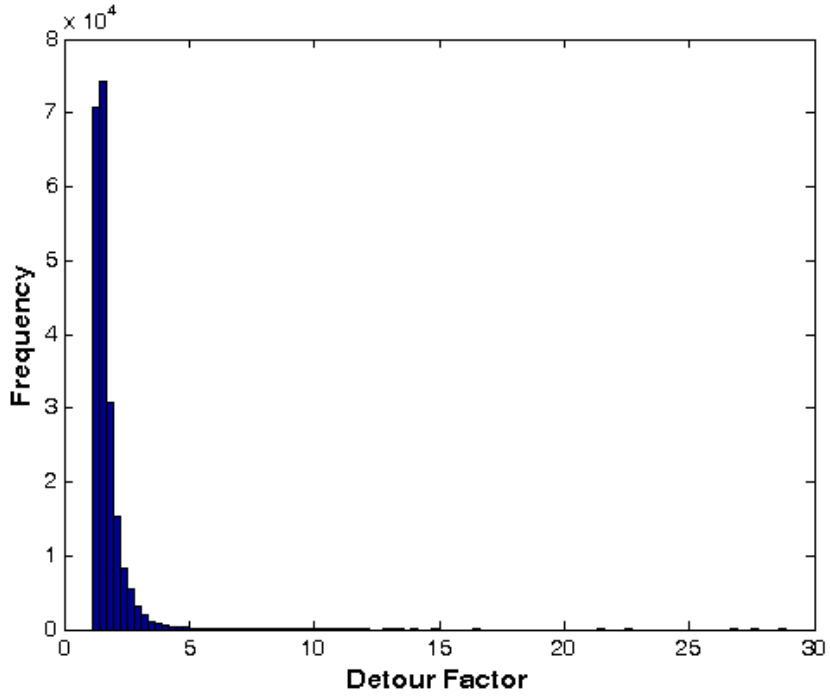


Figure C 26: Distribution of Detour Factors for Acela 110.

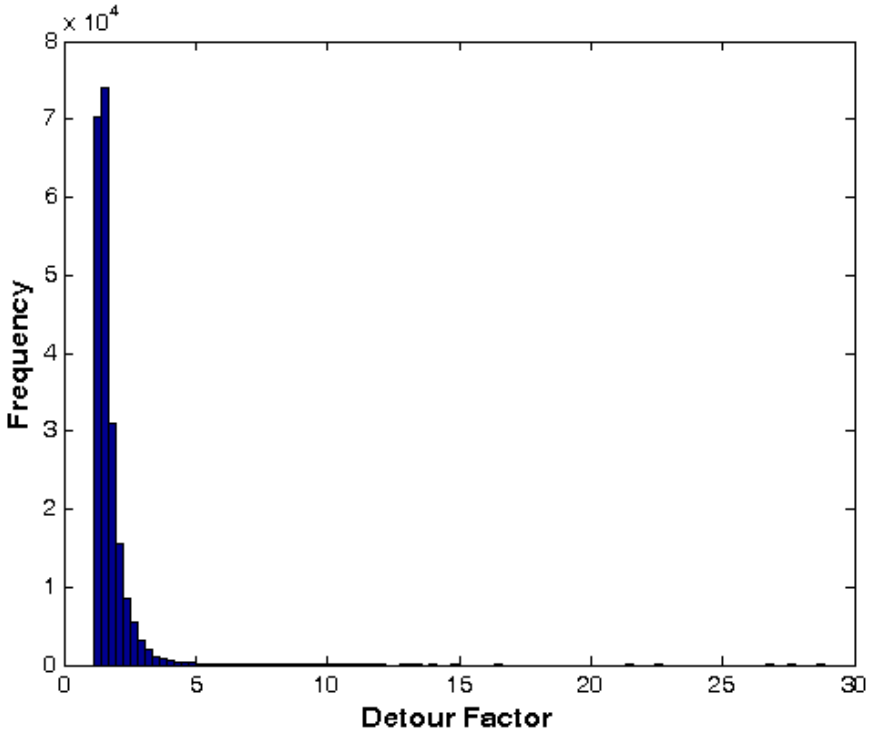


Figure C 27: Distribution of Detour Factors for Acela 125.

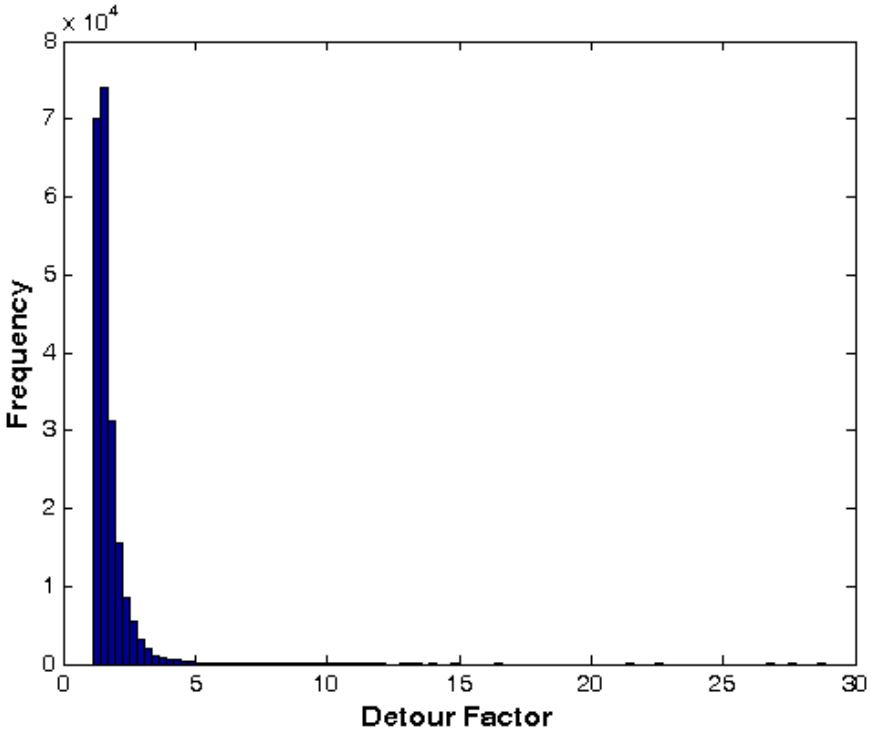


Figure C 28: Distribution of Detour Factors for Acela 150.

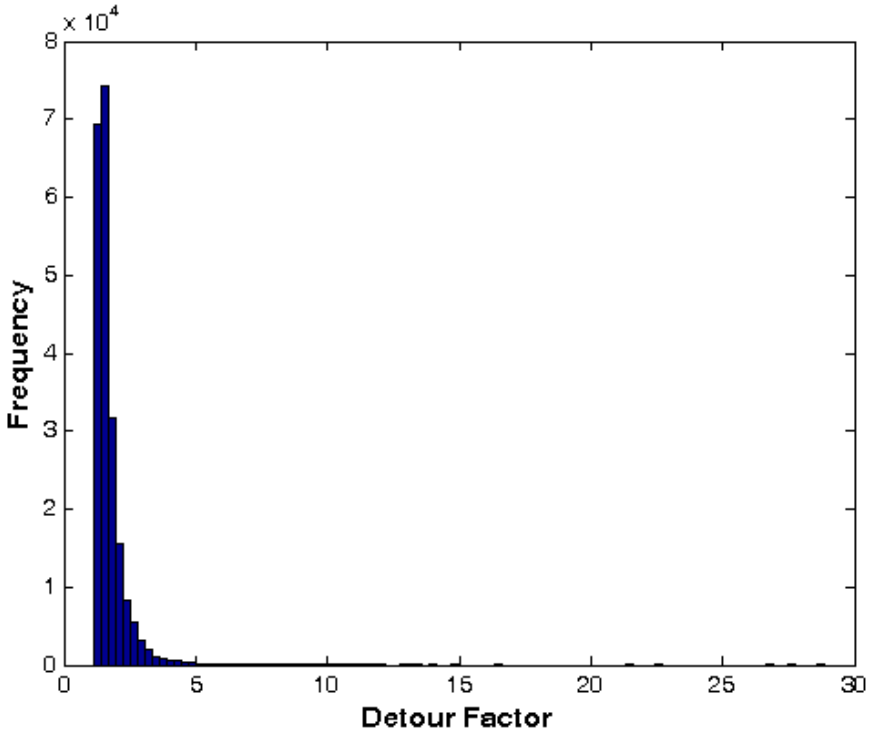


Figure C 29: Distribution of Detour Factors for Acela High Speed Rail.

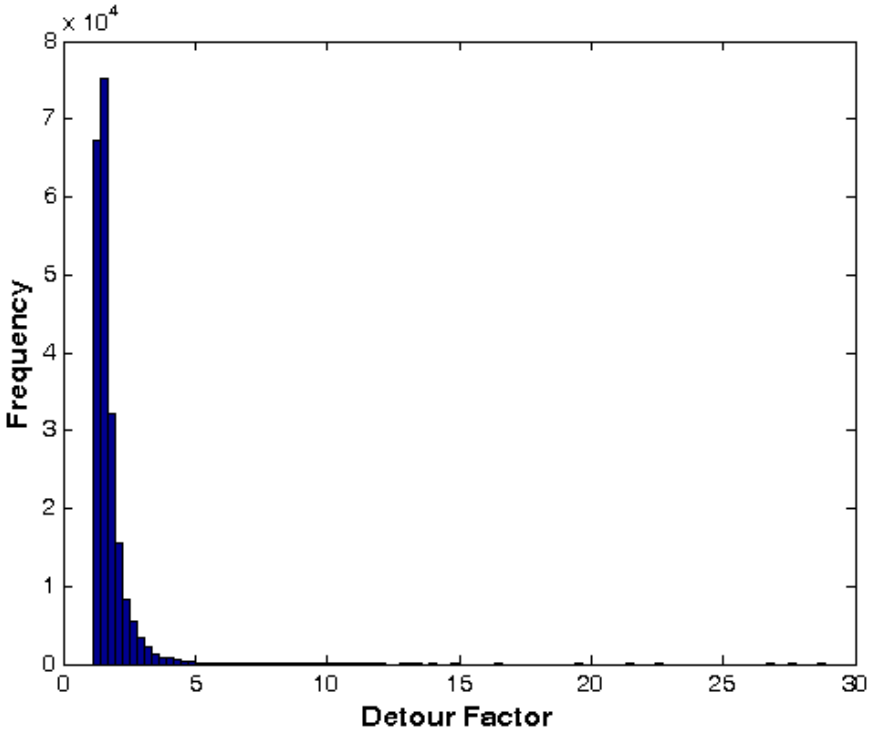


Figure C 30: Distribution of Detour Factors for Maglev.